3-26-2020 10:00 AM

Hyperspectral Image Classification for Remote Sensing

Hadis Madani
The University of Western Ontario

Supervisor
McIsaac, Kenneth
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Abstract

This thesis is focused on deep learning-based, pixel-wise classification of hyperspectral images (HSI) in the field of remote sensing. Although presence of many spectral bands in an HSI provides a valuable source of features which favors classification methods, dimensionality reduction is often performed in the pre-processing step to reduce the correlation between bands and spectral dimension of the input HSI. Most of the deep learning-based classification algorithms use unsupervised dimensionality reduction methods such as principal component analysis (PCA) which does not consider class labels. In this thesis, in order to take advantage of class discriminatory information in the dimensionality reduction step as well as the power of deep neural network in extracting abstract, deep features of HSI datasets, we propose a new method that combines a supervised dimensionality reduction technique, principal component discriminant analysis (PCDA) and deep learning. One common problem in remote sensing HSI classification is the lack of enough reliable ground truth samples. One solution to this dilemma can be data augmentation where virtual samples are generated from the ground truth examples. In this thesis, we propose a simple spectral perturbation method to augment the number of available training samples and improve the classification results.

Since combining spatial and spectral information for classifying hyperspectral images can dramatically improve the performance, in this thesis we also propose a new spectral-spatial feature vector. In our feature vector, based on their proximity to the dominant edges in the HSI, neighbors of a target pixel have different contributions in forming the spatial information. To obtain such a proximity measure, we propose a method to compute the distance transform image of the input HSI. We then improved the spatial feature vector by adding extended multi attribute profile (EMAP) features to it. Classification accuracies demonstrate the effectiveness of our proposed method in generating a powerful, expressive spectral-spatial feature vector.

Keywords: Remote sensing, hyperspectral image (HSI) classification, machine learning, stacked autoencoder (SAE), deep learning, supervised dimensionality reduction,

spectral-spatial features, data augmentation, distance transform, extended multi-attribute profile (EMAP), neural network (NN).

Summary for Lay Audience

In this thesis, we propose a few approaches to perform hyperspectral image (HSI) classification in the field of remote sensing. As opposed to the regular RGB images which consist of three channels of red, green, and blue, an HSI is composed of a series of images each taken at a specific wavelength. In the field of remote sensing, hyperspectral images are collected by the imaging sensors on board of an airplane or a satellite and due to the valuable information that they can provide about the objects and phenomena on our planet, they are employed in many applications.

One common practice in HSI processing is the classification of each individual pixel in the image. In other words, in many applications we are interested in assigning each pixel to a specific category. This kind of classification has been an active research area for years for which different approaches have been proposed. Recently, deep learning-based methods have attracted a lot of attention from the research community because of their superior performance compared to the conventional methods. Therefore, in this thesis, we used one of the deep learning frameworks, stacked autoencoder (SAE) to perform the pixel-wise HSI classification task.

As our first contribution, we combined SAE with a supervised dimensionality reduction (DR) technique where labels of the samples are used during the DR step. Second, as one of the common issues in processing the remote sensing hyperspectral datasets is the lack of enough labeled data, we proposed a method to generate virtual samples using the available ground truth data. Since combining spectral and spatial information in an HSI, can dramatically improve classification accuracies as our third contribution, in this thesis project we proposed a new method including a novel spectral-spatial feature vector. In our spatial feature vector, effective pixels have different contributions based on an edge proximity measure obtained from the distance transform image of the input HSI. We applied our methods on several remote sensing hyperspectral datasets and evaluated their performance using various accuracy metrics. Classification results show the superiority of our methods compared to several conventional and deep learning-based approaches.

Dedication

To my parents,

and my husband,

for their endless love and generous support.

Acknowledgments

I would like to express my sincere gratitude to Dr. Kennth McIsaac for his excellent supervision, immense knowledge, and continuous encouragement throughout the course of this research. It has been a great privilege and honor to pursue my higher education under his supervision.

I would like also to acknowledge Prof. J. Wang and Dr. B. Feng for providing us the Surrey dataset along with the ground truth image.

Thanks as well to the team members of the research group for their kindness, valuable feedback and helpful discussions.

Contents

List of Figures

List of Tables

List of Abbreviations

AA	Average accuracy
AE	Autoencoder
AI	Artificial intelligence
ANN	Artificial neural network
AP	Attribute profile
ASTER	Advanced spaceborne thermal emission and reflection radiometer
AVIRIS	Airborne visible infrared imaging spectrometer
BP	Backpropagation
CC	Connected component
CK	Composite kernel
CNN	Convolutional neural network
DAE	Deep autoencoder
DBN	Deep belief network
DR	Dimensionality reduction
ELM	Extreme learning machine
EM	Electromagnetic
EMAP	Extended multi-attribute profile
EMP	Extended morphological profiles
FC	Fully connected
GBN	Group belief network
GD	Gradient descent
GPU	Graphical processing unit
HAB	Harmful algal bloom
HISUI	Hyperspectral imager suite
HSI	Hyperspectral imaging
HyspIRI	Hyperspectral infrared imager
ISS	International space station
JPL	Jet propulsion laboratory

KNN	K-nearest neighborhood
Landsat	Land remote sensing satellite
LDA	Linear discriminant analysis
LSTM	Long short term memory
METI	Ministry of economy, trade, and industry
MLR	Multinomial logistic regression
MP	Morphological profile
MSE	Mean squared error
MSI	Multispectral imaging
NN	Neural network
OA	Overall accuracy
PC	Principal component
PCA	Principal component analysis
PCDA	Principal component discriminant analysis
RBF	Radial basis function
RBM	Restricted Boltzmann machine
ReLU	Rectified linear unit
RNN	Recurrent neural network
ROSIS	Reflective optics system imaging spectrometer
SAE	Stacked autoencoder
SE	Structuring element
SGD	Stochastic gradient descent
SVM	Support vector machine
SWIR	Shortwave infrared
TIR	Thermal infrared
VHR	Very high resolution
VSWIR	Visible to short wave infrared

Chapter 1

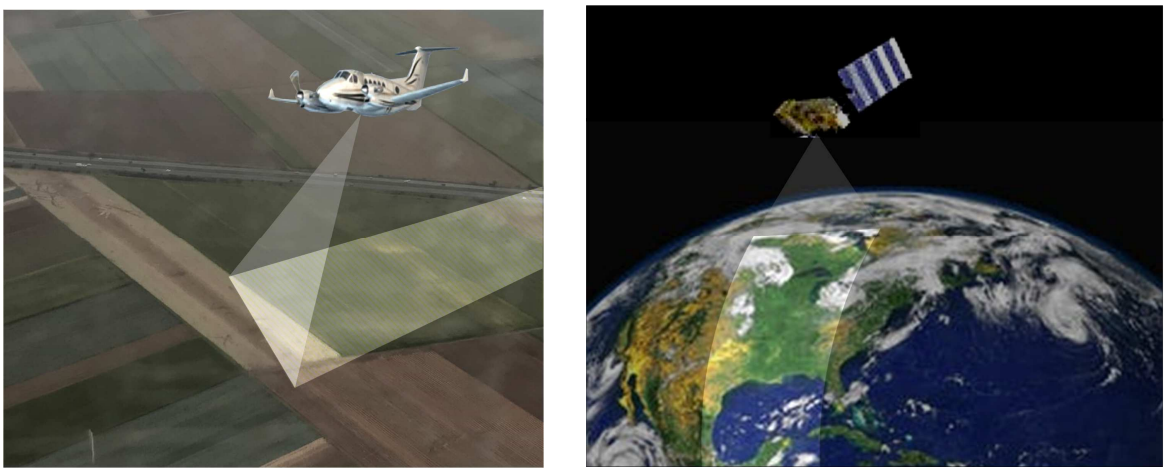
Introduction

This work presents new approaches in the field of remote sensing image analysis. Remote sensors provide a global perspective and a tremendous amount of valuable data about the Earth that could hardly been collected otherwise. The availability of such information lets scientists study the state of our planet and therefore make knowledge-based decisions. Powerful analysis of remote sensing data requires new techniques and approaches that produce meaningful results giving a comprehensive understanding of the objects and phenomenon on Earth. In this study, we address the problem of remote sensing hyperspectral image (HSI) classification using deep learning-based approaches that automatically produce feature representation of the hyperspectral datasets and boost the classification accuracies.

1.1 Remote sensing

1.1.1 Definition and applications

Remote sensing is the data acquisition process of a target or phenomenon in the absence of actual physical contact and allows us to acquire data from inaccessible and possibly



(a) (b)

Figure 1.1: Two main types of remote sensing. (a) Airborne and (b) space-borne remote sensing.

unsafe regions. It is used in most earth observation areas where it generally refers to the utilization of satellite- or aircraft-based sensor technologies to detect and classify objects on Earth. Therefore, the two main types of remote sensing includes airborne and space-borne remote sensing as shown in Figure 1.1. As can be seen from this figure, in the former, remote sensing data is obtained using the remote sensor on board of an airplane while in the latter, a satellite carries the imaging equipment. A remote sensor is an equipment that detects electromagnetic energy, measures it, and usually register it in an analogue or digital way. Remote sensors can be divided into two categories: passive and active. Figure 1.2 shows the schematic of these two types of remote sensors.

Passive remote sensors measure the energy that is naturally available. In the earth observation applications, the majority of the passive remote sensors detect the solar energy reflected back from the scene while the others measure the earth's emitted energy. There are caveats related to these two types of passive sensors. Solar energy dependent-sensors fail to operate at conditions where there is not enough or is no sunlight such as night times, in regions of the world permanently covered under the clouds or parts of the globe where sun's elevation is very low for most of the seasons resulting in unfavorable long shadows. On the other hand, the second type of passive sensors have difficulties

detecting earth's emitted energy since this energy corresponds to the low frequency-waves carrying low energies which makes them hard to be detected. A camera which is used to take pictures in the sunlight (i.e., flash of the camera is not used) is a simple example of a passive sensor.

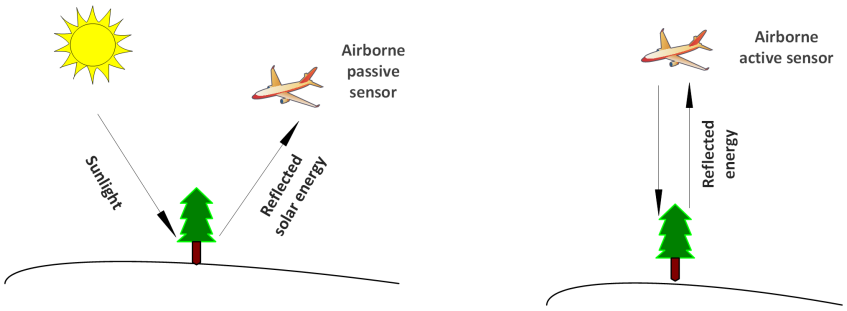


Figure 1.2: Active versus passive remote sensor. In passive remote sensing, Sun is often the source of energy whereas in active remote sensing, the airplane/satellite carries the energy source.

Different from the passive sensors, active sensors use their own source of energy for illuminating the object or scene of interest. In this case, electromagnetic energy is emitted through an energy source inside an airplane or a satellite and reflected radiation is measured by the sensor. Active sensors can then be used in both day and night, do not depend much on the weather conditions and provide a controlled light signal. Moreover, active sensors are used to measure the reflectance at the wavelengths that are not adequately provided by the sun such as microwaves. A common camera which is used in a relatively dark room and uses its own flash (energy source) to illuminate the scene is a simple example of an active sensor. The most commonly used active sensors for collecting data in the earth observation domain (geospatial data) include radar [1], laser fluorosensor and a synthetic aperture radar (SAR).

Examples of remote sensing applications include military surveillance, deforestation observation [2–4], glacier change monitoring [5,6], agriculture [7], waste management [8], etc. The following are few specific examples of the usage of remotely sensed images of Earth [9]:

- Huge forest fires can be viewed from the space which enables rangers to see a bigger portion of the ground under the fire; so, making more effective plans compared to when the fire is observed only from the ground.
- Weather forecasting by the means of cloud tracking
- Watching for the erupting volcanoes
- City growth monitoring and tracking the changes in certain farmlands for years or even decades
- Ocean floor mapping
- Counting polar bears in satellite images to ensure sustainable population levels
- Marine life preservation by detection of the oil spills and predicting their movement directions.

1.1.2 Remote sensing mission examples

A lot of effort have been made in the last two decades to collect remote sensing data widely used in the Earth science applications. Hyperspectral infrared imager, HyspIRI, mission conducted by NASA (2007-present) for instance, studies the existing ecosystems on our planet and provides with precious information about ecosystem changes and natural catastrophes such as volcanoes, wildfires, and drought [10]. In order to meet its goals, the HyspIRI data acquisition system is equipped with a visible-to-short-wave-infrared (VSWIR) imaging spectrometer with the spectral range of 380-2510 nm with 10-nm spectral resolution and a multispectral imager ranging from 3-13 μm containing 8 discrete bands covering the mid and thermal infrared (TIR) part of the electromagnetic (EM) spectrum. According to its final report in 2018, there are many potential applications related to the hyper and multi-spectral data obtained from the HyspIRI including but not limited to the following:

Catastrophes: Quantifying the possible dangers caused by volcanoes and wildfires using the data collected by the TIR sensor. Since the TIR instrument works at the range of 3-13 μm and because of the fact that very hot objects emit energy at the wavelength of 4μm and thanks to the high spatial resolution of the TIR sensor, it is easy to detect pixels that correspond to active lavas or active fire burns (e.g. wildfires). Moreover, due to the capability of this sensor to measure a wide range of energy intensities, it would be possible to identify the most active lavas/fires.

Water Quality: The HyspIRI mission provides the hyperspectral images ranging from the visible to shortwave infrared and multispectral thermal data that provides a valuable source of information enhancing the water quality monitoring. Observation of the optical characteristics of water by employing the hyperspectral imaging can be a powerful mean for water quality assessments especially for the water bodies that have been exposed to contaminants such as harmful algal blooms (HABs) [11] over the years. As a specific example of this application, we can name the HAB monitoring on the Great Lakes which is the largest source of freshwater worldwide and provides the drinking water for 40 million residents of the U.S. and Canada. The characteristics of the HyspIRI system (visible-shortwave infrared and thermal wavelengths) help improve the identification of the toxic bacteria related to HABs and also their spatial distributions through the Great Lakes' surface water.

Another example of the missions with the goal of collecting hyperspectral data from the Earth is the hyperspectral imager suite (HISUI) mission [12]. HISUI is a spaceborne hyperspectral imaging system which has been established by the Japanese ministry of economy, trade, and industry (METI) as its forth spaceborne optical imaging project beginning in the year 2006 and scheduled to be launched for January 2020 through SpaceX's Falcon-9 [1] (SpX-20) to be deployed on the international space station (ISS). One of the mentioned potential functionality of the manufactured instrument (hyperspectral imager) will be applications such as oil resource exploration. The imaging system of

[1]Falcon 9 is a two-stage-to-orbit medium lift launch vehicle designed and manufactured by SpaceX in the United States.

HISUI includes a reflective telescope and two spectrometers working in the near infrared and shortwave infrared (SWIR) regions of the EM.

HyspIRI and HISUI missions are just two examples of the many projects carried out in the area of remote sensing hyperspectral imaging. More projects are expected to be planned for the future due to the need of the human being to have more information about the phenomena occuring on our planet by the means of the irreplaceable data that spaceborne or airborne hyperspectral imaging systems can provide. The effort and cost of the missions such as HyspIRI and HISUI delivering the valuable source of information is well appreciated only in the presence of powerful hyperspectral and multispectral image analysis approaches. Currently, deep-learning based methods are the state of the art algorithms for classification of such data and even though there have been a large number of researches in this realm, there is still a lot to be explored.

1.2 Research problem

Motivated by the countless number of applications related to remote sensing hyperspectral images and the applicability of the artificial intelligence (AI) in processing hyperspectral data, In this study, we performed experiments on four hyperspectral image databases and proposed three new deep-learning based methods to carry out pixel-wise classification of these datasets.

1.2.1 Hyperspectral image classification

A typical hyperspectral image is composed of many images each taken at a specific wavelength. So, it can be imagined as a cube where the length and width of the cube corresponds to the spatial extent (number of pixels) of the 2-d image at each wavelength while its depth represents the number of spectral bands of the hyperspectral image. A typical hyperspectral dataset is shown in Figure 1.3. In HSI, imaging and spectroscopy

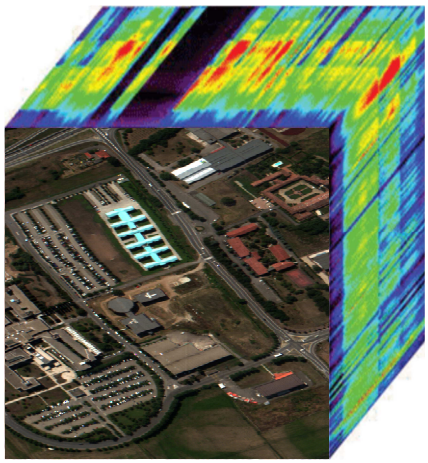


Figure 1.3: A sample hyperspectral image. This hyperspectral database was taken with Reflective Optics System Imaging Spectrometer (ROSIS) during a flight over Pavia University, in northern Italy.

are combined to obtain a great source of spectral and spatial information of the scene. Because of the fact that different material own different spectral signatures, in an HSI, the type of the objects at each pixel can be identified using the spectral information while the spatial data provides their spatial distribution in the image. Hyperspectral imaging is explained in more detail in Section 2.1.

There are problems associated with hyperspectral image classification especially using deep neural networks: The large spectral size of the database which introduces a lot of tunable parameters to the model and the unavailability of adequate ground truth data to train the model effectively. Performing dimensionality reduction and data augmentation are two category of approaches to reduce the effect of these bottlenecks. In Chapters 3 and 5, we will propose two methods to deal with these problems. Moreover, since one important factor in HSI classification is to use the spatial information as effectively as possible, in Chapter 4, we propose a novel approach to extract spatial information to boost the classification accuracies.

The conventional and new approaches in hyperspectral image classification are described in Section 2.2.

1.3 Research contributions

This thesis is divided in six chapters and includes the following contributions:

- A new technique for hyperspectral image classification based on a combination of a supervised data dimensionality reduction method and a deep learning framework is proposed which results in high classification accuracies and smaller testing time.
- A new deep learning-based technique to perform pixel-wise classification of remote sensing hyperspectral scenes based on a novel spatial feature representation is proposed. We applied this proposed method on a new hyperspectral dataset, Surrey. This method improves the classification accuracies and the test time.
- A new approach for computing the distance transform image from an input hyperspectral image is proposed.
- A simple new technique for augmenting the available ground truth data using a spectral perturbation method is proposed.
- A through search in the models' hyperparameter space was performed to find the optimum values for these quantities.

Chapter 2

Background and Literature Review

2.1 Background

Spectral imaging for remote sensing of the ground's objects and features have become an active field of study among researchers. This technology as an alternative to the high-spatial resolution, large aperture satellite imaging systems has brought ease and convenience in the remote sensing domain. To have a better understanding of this technology lets first see what the electromagnetic (EM) spectrum is.

EM spectrum is the term that describes the whole range of EM radiation. EM radiation can be represented by the means of waves or photons. Based on the wave theory, unless influenced by an outside object, light travels in a straight line and the energy it carries oscillates in a wave fashion. The two oscillating components of light include electrical energy and magnetic energy. A schematic of the EM spectrum is shown in Figure 2.1. As can be seen from this figure, EM spectrum is composed of different types of EM radiation including radio waves, microwaves, infrared light, visible light, ultraviolet light, X-rays and gamma-rays. In fact, visible light is the only part of the EM spectrum which can be sensed by human eyes and it only covers the small wavelength range between about 400 nm to 750 nm.

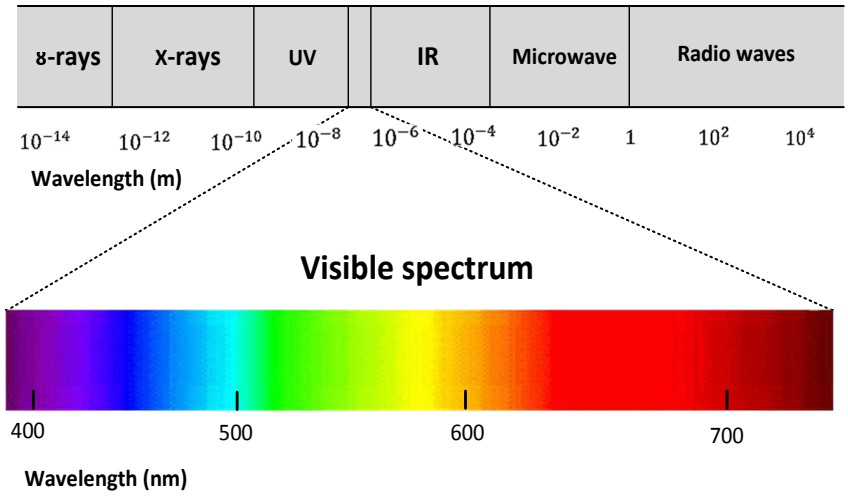


Figure 2.1: Electromagnetic spectrum

In the early applications of spectral imaging, only a small number of selected bands in the visible and infrared regions of the electromagnetic spectrum was used which in this case, it was called multi spectral imaging (MSI). Advanced spaceborne thermal emission and reflection radiometer (ASTER) [13] and land remote sensing satellite (Landsat) [14] are two of the most well-known MSI systems. In the newer version, HSI, hundreds of contiguous spectral bands are employed to identify various natural and human manufactured materials. Visible light and infrared radiation are the most commonly used regions of EM spectrum in remote sensing applications. Airborne visible/infrared imaging spectrometer (AVIRIS) [15], designed by NASA at jet propulsion laboratory (JPL) in 1980s, is an excellent instance of an HSI system. Because of the valuable amount of information that HSI datasets can provide, they are used in many areas such as remote sensing [16–19], agriculture [20], food processing [21–23], face recognition [24], etc.

The concept of hyperspectral imaging is that for different materials the value of radiation that is reflected, absorbed, or emitted is a function of the wavelength. In hyperspectral imaging sensors, for each square pixel area in the scene composing of various materials and for a large number of consecutive spectral bands, the amount of the radiance is measured. Figure 2.2, shows the concept of hyperspectral imaging. Four major components of any remote sensing hyperspectral imaging system includes: the

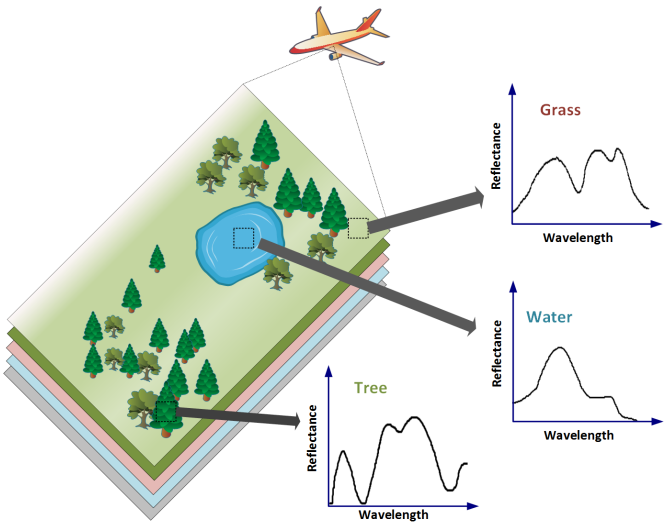


Figure 2.2: Concept of hyperspectral imaging. A large area on the ground is being imaged by an airborne/spaceborne imaging device covering a wide range of electromagnetic spectrum. Different reflectance values for the three sample materials in the scene depicts how different classes can be identified using their reflectance values.

radiation (or illuminating) source, the atmospheric path, the imaged surface, and the sensor [25]. In passive remote sensing, where the sun is often the main illumination source, what is measured by the sensor is the solar energy that has emitted from the sun, traveled through the atmosphere, interacted with materials on the earth's surface and reflected back to the sensor. At this point, this measured energy is transformed into a digital form for further processing. The reflectance spectrum, or spectral signature of any material, is a function of the wavelength λ and is defined by (2.1)

$$\text{reflectance spectrum } (\lambda) = \frac{\text{reflected radiation at band } (\lambda)}{\text{incident radiation at band } (\lambda)} \tag{2.1}$$

where the numerator is the reflected energy by the material and the denominator represents the incident energy (energy received by the material) at different wavelengths [25]. It should be noted that solar energy is absorbed by the oxygen and water vapor in the atmosphere at some specific wavelengths, called absorption bands, owning very poor signal to noise ratio. Therefore, in real applications, these bands are discarded.

Although we do not need hundreds of spectral bands spread over a wide range of

electromagnetic wave to detect a single material uniquely, in real conditions where that single material is combined with other materials on the earth surface and being imaged through severe changeable atmosphere, it would be better to have many bands (features) than a few. In fact, having a large number of spectral bands makes it possible to apply statistical methods on the data composing of pixels each containing various material components [26].

2.2 Literature review

Since remotely sensed hyperspectral images can cover wide areas on the ground as well as provide reflection information at so many spectral bands, they provide a wealth of data for researchers and scientists. A lot of effort have been made to utilize the information hidden in remote sensing hyperspectral images as effectively as possible employing machine learning-based techniques. Traditional classifiers such as support vector machine (SVM) classified each pixel using only its spectral information [27–29]. In other words in these methods, spectrum of each pixel was given to the classifier as the input feature vector. K-nearest neighborhood (KNN) and its variations are another types of HSI classification methods using only spectral information as the pixels' features [30]. Researches depict that spectral information provide a useful source of information to perform the classification task with reasonable amount of accuracy. However, since the adjacent pixels in an HSI share similar spectral characteristics, combining the spectral and contextual spatial information help classify pixels with higher degrees of accuracy.

Early works on combining spectral and spatial information to classify spectral imagery data was devoted to multispectral images [31, 32]. Later, Pesaresi *et al* proposed a new method to incorporate spatial information using morphological profiles (MPs) [33]. MPs of a gray-level image are obtained by applying a set of morphological operations called opening and closing by reconstruction by a structuring element (SE) of fixed shape and increasing size on the image such that some spatial details in the image are weakened while

some other are maintained. An extension of the MP called extended morphological profile (EMP) was proposed by Benediktsson *et al* [34] and Fauvel *et al* [35] to generalize the idea to multi/hyper spectral images. In EMP, first principal component analysis (PCA) is applied on the input hyperspectral image to reduce the dimensionality of the data as well as band correlations. Then the MP method is applied on the first few principal components (PCs). Eventually, the MPs of all PCs are stacked together forming the final EMP structure. Definition of PCA, MP, and EMP are presented in Section 2.3.1, 2.3.5.1, and 2.3.5.2, respectively.

Although EMP could successfully model spatial information of a hyperspectral image, it had some drawbacks such as inability to model various spatial information due to the fixed shape of the SE in obtaining the MPs. Attribute profiles (APs) as an improvement to MPs was proposed by Dalla Mura *et al* [36] in 2010. APs of an image are a set of profiles obtained by applying some attribute filters on the image. These attribute filters process the input image at different levels and by removing connected components (CCs) that do not satisfy a criterion related to the attribute. An attribute in that sense can be any measure computable on the connected components of the image. For example, the size of the connected components can be an attribute. APs are more powerful than the MPs in modeling the spatial information of an image since they process the input based on different attributes flexible in their definitions. Similar to EMP, the extension of APs was proposed by Dalla Mura *et al* [37] called extended attribute profile (EAP) to make it applicable on hyperspectral datasets. In the case of employing multiple attributes the structure is called extended multi attribute profile (EMAP) [37]. Explanation of AP, EAP, and EMAP are given in Sections 2.3.5.3, 2.3.5.4, and 2.3.5.5, respectively.

Another class of techniques which combines spectral and spatial information together are composite kernel (CK) methods. In this type of methods, spectral and spatial information of a target pixel are combined through kernel functions. In [38], authors considered some statistical measurements of the neighboring pixels of a target pixel such as their mean or standard deviation values at each spectral band as its spatial information. Then, they combined spectral and spatial data by the means of a family of

CKs satisfying the Mercer's conditions and used SVM classifier on top to perform the classification. A new set of generalized CKs was proposed in [39] to combine spectral and spatial information together with no weight parameters. In order to compute the spatial feature vector prior to apply the kernel function, the authors used EMAP data structure and finally used the multinomial logistic regression (MLR) classifier to perform the classification. In [40], CKs along with extreme learning machine (ELM) is used to perform classification of HSI datasets employing joint spectral-spatial features.

Although all of the mentioned spectral-spatial feature extraction techniques could successfully deliver a representation of the two kinds of information existing in the hyperspectral datasets, they are extremely hand-crafted. For example, in EMAP method, user needs to identify what type of attributes he wants to use. Neural network (NN) has found its way to solve regression or classification problems in many areas specifically image classification where HSI classification was no exception. In fact, a lot of researches in recent years concentrated on employing NN as an automatic feature representation technique in classification of hyperspectral images. To speak more specifically, it is deep neural network (DNN) that has been considered as the most popular technique for HSI classification (or any type of image classification in general) in the past few years thank to the development of powerful graphical processing units (GPUs) and the availability of more training data. Recently, several studies in the remote sensing field have used deep learning models in order to perform hyperspectral image classification [41–48]. Results of these studies demonstrated superior performance of deep learning methods in hyperspectral image classification compared to the conventional approaches.

In [41] and [49], for the first time the concept of deep learning was used for hyperspectral image classification. In [41], stacked autoencoder (SAE) was used as the deep network to extract deep features of each training and test pixel in the hyperspectral image. Three kinds of features were extracted and used for classification: spectral features, spatial-dominated features, and joint spectral-spatial features. In the case of spectral features, spectrum of each pixel is given to the network as input. In order to extract spatial-dominated features, first, PCA is applied on the whole hypercube to reduce the

spectral dimensionality. Then, a neighborhood around each pixel is determined and is converted in the form of a 1-D vector which will be the input of the network. Finally in the last case, spatial-dominated information of each pixel are concatenated to the spectrum of the pixel to form the joint spectral-spatial feature vector which is fed to the network. Having pre-trained the all layers in the SAE, in order to perform fine-tuning and classification, all layers are connected and a logistic regression classifier is put on the top of the network. The output results revealed that the proposed approach outperformed the state of the art methods used for HSI classification. Motivated by [41], in [44], a new feature learning method, called contextual deep learning (CDL) is proposed. Similar to [41], spectral and spatial features are extracted before classification. However, unlike [41], this method reduces the features' spectral dimensionality and extracts the spatial features at the same time. In order to perform classification, MLR was used.

In [43], in order to extract deep spectral-spatial information, an improved version of SAE called spatially updated deep auto-encoder was introduced. The first contribution of this study was altering the energy function of each auto-encoder to ensure that correlation between samples is preserved while encoding them. Next, in order to take spatial information into account, a feature updated layer is embedded after the hidden layer which replaces each feature with an average value of the features extracted from the pixels in the neighborhood of the target pixel. In order to deal with not having enough training samples and also smooth the classification result, the authors introduce collaborative representation based classification approach [50] into HSI classification domain to obtain an output vector (whose size equals to the number of classes) for each target pixel whose elements express the probability of belonging the pixel to each category. The final smoothed classification map was obtained by solving maximum *a posteriori* (MAP) probability segmentation problem [51]. In [48], EMAP features with sparse autoencoder are used to classify three very high resolution (VHR) multispectral datasets. They have used a one layer SAE and employed area (a) and the standard deviations of the pixels inside the connected components (s) as the attribute filters. Also, they used two threshold values for each attribute.

In [42, 46, 52], the task of hyperspectral image classification is performed using deep belief network (DBN) where restricted Boltzmann machine (RBM) is employed as the building block of the DBN. In [52], a DBN (with two hidden layers) extracts spectral-spatial features of the pixels in the hypercube such that first, the spectral dimensionality of the hypercube is reduced using PCA and only first three principal components are retained. Next, a 3D patch of size 7x7x3 around each training sample (pixel) is formed which was later vectorized and was given to the the network as input. Just like SAE, DBN is trained in a layer-wise manner. The final layer of the DBN consisted of a logistic regression (LR) classifier. In 2017, the concept of using grouped features was proposed by Zhou *et al.* [46]. Their method, called group belief network (GBN), adaptively diminishes the weights of the connections which correspond to the irrelevant spectral bands. Similar to DBN, the proposed GBN is constructed of stacked RBMs; however, the bottom layer of the DBN is replaced by a modified version of an RBM named as Group-RBM (GRBM). The GRBM has the capability of managing grouped features. Proposed GBN-based HSI classification method has been applied on three HSI datasets and compared to DBN the segmentation results have been slightly improved.

Some recent studies have used convolutional neural network (CNN) as the deep network structure in order to extract spectral and spatial information both in supervised and unsupervised manners [45,47,53–58]. [53] considers each sample vector (sample spectra) as a 2D image; therefore, the input of the network is the spectral signature of each pixel. Structure of the CNN used in this study consists of a convolutional layer (C1), a max-pooling layer (M2), a fully connected layer (F3), and the output layer. Although the resulting classification accuracies showed the capability of the CNN in hyperspectral image classification, maximum overall accuracy of 92.6% implies that results could be further improved. Unlike [53] which does not use spatial correlation between samples for classification of the hyperspectral data, in [45], Zhao *et al.* proposed spectral-spatial features by employing balanced local discriminant embedding (BLDE), an extension of LDE algorithm introduced in [59], for reducing the spectral dimension of the input data and a CNN for spatial feature extraction. In order to extract spatial features of pixels, first, the

dimension of the original data (hypercube) is reduced along the spectral dimension using BLDE and only first few principal bands are kept. Next, a squared patch around each training pixel is formed. These patches are used as training data set for training the CNN in a supervised manner. The features in the last layer of the CNN framework are flattened and form the spatial feature vectors. By denoting $\boldsymbol{z}_i$ as the spectral feature and $\boldsymbol{o}_i$ as the spatial feature of the unknown test sample $\boldsymbol{x}_i$ where the former feature is obtained using the BLDE method and the latter feature is computed by the trained CNN (on the training patches), the final feature for the test sample is obtained by concatenating these features as $[\boldsymbol{z}_i, \boldsymbol{o}_i]$. In 2017, Li et al. dealt with the small number of training samples by building a Pixel-Pair model using available training samples [47]. The procedure is as follows: having M training samples of C different classes and expressing each training sample as $\{\boldsymbol{x_i}, y_i\}$ where $\boldsymbol{x_i}$ is the training sample and y_i is its corresponding label, for increasing the number of labeled training samples any combination of two samples of all classes is randomly chosen and is called $\boldsymbol{S_{ij}}$ where $\boldsymbol{S_{ij}} = [\boldsymbol{x_i} \quad \boldsymbol{x_j}]$. If the two samples are drawn from the same class, $\boldsymbol{S_{ij}}$ will also have the same class label as theirs. However, if they belong to different classes, the label of 0 will be assigned to $\boldsymbol{S_{ij}}$. [47] resembles the approach proposed in [45] in that they both take advantage of spatial features as well as spectral information of the pixels. However, in [47] the incorporation of the spatial information is done in the test phase through introducing the *"Joint Classification With Voting Strategy"*. This voting strategy is based on the fact that neighboring pixels belong to the same class with a high probability. In the test step, each test sample is also combined with its neighbors to form pixel-pair samples and the target pixel will be assigned to the class to which the majority of its neighbors belong to.

In [54], in order to exploit spectral and spatial information, a 3D-CNN has been designed. This architecture can extract both spectral and spatial information simultaneously because it forms a small 3D window (patch) around the target pixel in the hypercube and feeds as input this 3D patch to the network. After several convolutional and pooling layers the extracted features will be in the form of a vector which contains deep spectral and spatial information of the target pixel. This network can have fewer

trainable parameters but the computation cost is highly increased due to the convolution along spectral bands. The approach used in [55] for extracting the spectral and spatial information simultaneously from the data hypercube is similar to [54]. In other words, a 3D patch (tensor) around each target pixel is formed which contains both spectral and spatial information. However, unlike [54] which feeds these 3D tensors to the CNN as input directly, in [55] Randomized PCA (R-PCA) is applied on each 3D tensor in order to reduce the dimensionality of the input data. The proposed CNN also differs from the CNN used in [54] and also from a conventional CNN in that there is no pooling layer in the structure of the CNN.

2.3 Machine learning and image processing

Machine learning as one of the sub-categories of artificial intelligence (AI), is the science of automatically finding the hidden patterns in the data without human interventions. The fundamental requirement for developing a machine learning-based model is some sample data called *training data* used to train the model. After the model is trained using the the training data, it will be able to make predictions or decisions on the new unseen data called *testing data*. The important concept of machine learning is that computers are not programmed explicitly to perform a task, rather are taught to learn for themselves and make decisions from what they have already seen (training data). Machine learning algorithms falls within the two main categories of supervised and unsupervised methods [60]. In supervised methods, training data includes data points with their labels while in unsupervised approaches labels are not given to the model during training. In the following subsections, we introduce some of the machine learning methods from both categories which we will refer to later in this thesis. Furthermore, in Section 2.3.5 we describe a morphological-based image processing technique used to extract spatial information from input images.

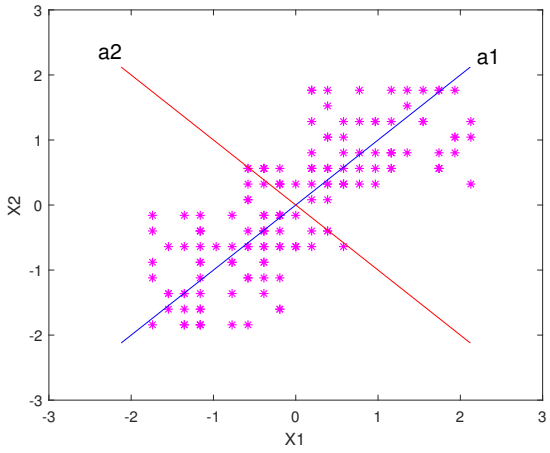


Figure 2.3: A toy example showing the directions of maximum variance in the data obtained by the PCA algorithm

2.3.1 Principal Component Analysis

PCA as a popular dimensionality reduction (DR) method has been introduced by Karl Pearson in 1901 [61] and has been used frequently in the literature in many fields ever since. This method searches for the most accurate data representation in a lower dimensional subspace composed of the uncorrelated linear combinations of the original variables called principal components. What PCA does is in fact, mapping the input data to the dimensions along which data vary the most; so, preserving the largest variances in the data. A toy example of applying the PCA algorithm on a sample of 2 dimensional points is shown in figure 2.3. As can be seen from this figure, $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ point to the directions of the largest and the second largest variances in the dataset, respectively.

Suppose we have a data set composed of N, p dimensional samples. We can show the aforementioned efficient linear combinations of the original variables (PCs) as follows

$$\begin{aligned} Z_1 &= a_{11}X_1 + a_{12}X_2 + ... + a_{1p}X_p = \boldsymbol{a}_1^T\boldsymbol{X} \\ Z_2 &= a_{21}X_1 + a_{22}X_2 + ... + a_{2p}X_p = \boldsymbol{a}_2^T\boldsymbol{X} \\ &\vdots \\ Z_p &= a_{p1}X_1 + a_{p2}X_2 + ... + a_{pp}X_p = \boldsymbol{a}_p^T\boldsymbol{X} \end{aligned} \tag{2.2}$$

where Z_i is the ith principal component, $\boldsymbol{a}_i$ represents the ith loading vector. Also, $\boldsymbol{X}$ is a p-dimensional vector representing a point in a p-dimensional space. In the PCA algorithm, these PCs are computed through the following process:

Define the first principal component of the sample $\boldsymbol{X} = (X_1, ..., X_p)$ by the linear transformation

$$Z_1 = \boldsymbol{a}_1^T\boldsymbol{X} = \sum_{i=1}^{p} a_{1i}X_i \tag{2.3}$$

where the vector $\boldsymbol{a}_1$ is chosen such that $var\,(Z_1)$ is maximized.

Similarly define the kth PC of $\boldsymbol{X}$ according to (2.4)

$$Z_k = \boldsymbol{a}_k^T\boldsymbol{X} = \sum_{i=1}^{p} a_{ik}X_i \qquad k = 1, .., p \tag{2.4}$$

where vector a_k is chosen such that the following conditions are met:

1. $var\,(Z_k)$ is maximized

2. $cov[Z_k, Z_l] = 0 \quad$ for $\quad k > l \geq 1$

3. $\boldsymbol{a}_k^T\boldsymbol{a}_k = 1$

The above conditions leads to following properties of PCA:

- $var(Z_1) \geq var(Z_2) \geq ... \geq var(Z_p) \geq 0$
- Uncorrelated PCs and orthogonal loading vectors, $\boldsymbol{a}_k$:

$cov[Z_l, Z_m] = 0 \quad$ and $\quad \boldsymbol{a}_l^T \boldsymbol{a}_m = 0$ for $l \neq m$

- Preservation of the total variance:
  $\sum_{i=1}^{p} var(Z_i) = \sum_{i=1}^{p} var(X_i)$

The last property notes that there are the same amount of variance (information) in the whole set of PCs as there are in the original set of data. However, since we are interested in reducing the dimensionality of the input data, we only keep the first few PCs which contain the most information of the original data. It has been mathematically proven that the loading vectors that meet above requirements are in fact, the eigenvectors of the covariance matrix of the original points in the dataset [62].

Let's show the eigenvalues and the corresponding eigenvectors of such a covariance matrix with λ_i and $\boldsymbol{e}_i$, respectively, where i is the index of these eigenvalues/vectors. Considering equation (2.2) and the fact that the desired loading vectors in this equation are the eigenvectors $\boldsymbol{e}_i$, we can rewrite them as follows:

$$\begin{aligned} Z_1 &= e_{11}X_1 + e_{12}X_2 + ... + e_{1p}X_p = \boldsymbol{e}_1^T \boldsymbol{X} \\ Z_2 &= e_{21}X_1 + e_{22}X_2 + ... + e_{2p}X_p = \boldsymbol{e}_2^T \boldsymbol{X} \\ &\vdots \\ Z_p &= e_{p1}X_1 + e_{p2}X_2 + ... + e_{pp}X_p = \boldsymbol{e}_p^T \boldsymbol{X} \end{aligned} \tag{2.5}$$

Also, it can be proven that for each Z_k:

$$var(Z_k) = \lambda_k = \boldsymbol{e}_k^T \boldsymbol{S} \boldsymbol{e}_k \tag{2.6}$$

where $\boldsymbol{S}$ is the covariance matrix of the variables of the original dataset. In other words, the kth largest eigenvalue of $\boldsymbol{S}$ is the variance of the kth PC and the kth largest fraction of the variation in the points of our dataset is preserved by this PC. It should be noted that in many applications, the true covariance matrix is not available so, what $\boldsymbol{S}$ represents in these equations is the sample covariance matrix of the set of given data points formed

in an $N \times p$ matrix. In equation (2.5), Z_1 to Z_p are the p PCs of one observation in the dataset (one row of the matrix of the input data of size $N \times p$). Equation (2.7) can be used to compute the PCs for all N observations in the dataset:

$$\boldsymbol{Z}_{N\times p} = \boldsymbol{X}_{N\times p}\boldsymbol{E}_{p\times p} \tag{2.7}$$

where each row of matrix $\boldsymbol{X}$ is an observation of size $1 \times p$, $\boldsymbol{E}$ is a $p \times p$ matrix whose columns contain the p (normalized) eigenvectors of the covariance matrix $\boldsymbol{S}$, and $\boldsymbol{Z}$ is the PC matrix whose jth row contain the PCs of the jth observation in $\boldsymbol{X}$.

2.3.2 Linear Discriminant Analysis (LDA)

The most common usage of LDA is dimensionality reduction. This method was first introduced by Ronald A. Fisher in 1936 [63] as the feature extraction step for discriminating between two classes of flowers. In 1948, a generalized version of this method called multi-class "Linear Discriminant Analysis" or "Multiple Discriminant Analysis" was introduced by C. R. Rao for a multiple class problem [64]. Although this DR method is very similar to the PCA algorithm, LDA is a supervised approach. In other words, we use class labels while reducing the dimension of the data and try to preserve as much of the class discriminatory information as possible. In that sense, the main objective of LDA method is projecting data into a lower dimensional subspace such that class-separability gets maximized. Figure 2.4 shows a toy example of the difference between PCA and LDA algorithm for two-dimensional data divided into two classes. As can be seen from this figure, PCA projects data into the direction of the maximum variance in the data regardless of their labels wheres LDA tries to find direction of the most variation of the data points while maintaining class separability at the same time.

Considering $\boldsymbol{X}$ as a dataset consisting of N p-dimensional samples with N_i samples in each of the C classes each denoted by w_i, LDA searches for a transformation function that maps data in $\boldsymbol{X}$ to a $C-1$ dimensional subspace $\boldsymbol{y}$ while keeping the most possible

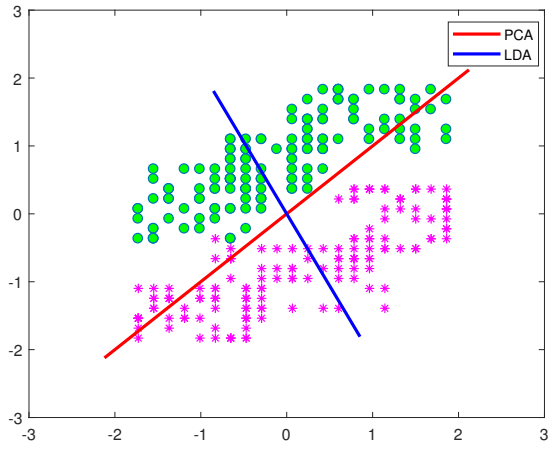


Figure 2.4: A toy example showing the best projection directions suggested by PCA and LDA algorithms.

class discriminatory information of the data. Let's start with the two class problem and then generalize it to the C-classes case.

2.3.2.1 Two-classes case

Let's say we have a database of p-dimensional samples $\boldsymbol{x} = [x_1, x_2, ..., x_p]^T$, where N_1 and N_2 of those samples belong to classes w_1 and w_2, respectively. The objective is to find the transformation $\boldsymbol{a}$ in (2.8) that maps data from the original space $\boldsymbol{x}$ into the lower-dimensional subspace $\boldsymbol{y}$ while preserving the maximum class separability in this new subspace.

$$y = \boldsymbol{a}^T \boldsymbol{x} \tag{2.8}$$

where $\boldsymbol{a} = [a_1, a_2, ..., a_p]^T$.

Since we have a two-class problem (i.e., C=2), y will be a one-dimensional space, so, each original vector $\boldsymbol{x}$ will be mapped to a scalar. In order to find $\boldsymbol{a}$, we need to specify a separability measure between the projected samples. The approach proposed by Fisher [63] was to maximize the ratio of the difference of the two classes' means (between-class variance) normalized by a term which is a function of the within-class variation,

called scatter according to 2.9

$$J(\boldsymbol{a}) = \frac{\left|\widetilde{\mu}_1 - \widetilde{\mu}_2\right|^2}{\widetilde{s}_1^2 + \widetilde{s}_2^2} \tag{2.9}$$

where $\tilde{\mu}_i$ and $\widetilde{s}_i^2$ are the mean and the within-class variation of the samples of class w_i having been mapped to the new subspace y, respectively. $\widetilde{\mu}_i$ can be calculated according to (2.10)

$$\widetilde{\mu}_i = \frac{1}{N_i}\sum_{y\in w_i} y = \frac{1}{N_i}\sum_{\boldsymbol{x}\in w_i} \boldsymbol{a}^T\boldsymbol{x} = \frac{\boldsymbol{a}^T}{N_i}\sum_{\boldsymbol{x}\in w_i} \boldsymbol{x} = \boldsymbol{a}^T\boldsymbol{\mu}_i \tag{2.10}$$

where $\boldsymbol{\mu}_i$ is the mean of samples in class w_i in the original p-dimensional space. In fact, the numerator in (2.9), states how far the mean of the projected samples in the two classes are from each other. The equation for each class's scatter is the same as its variance

$$\widetilde{s}_i^2 = \sum_{y\in w_i} \left(y - \widetilde{\mu}_i\right)^2. \tag{2.11}$$

We can define $\widetilde{s}_1^2 + \widetilde{s}_2^2$ as a measure of the within-class variation of the projected samples and it is called the within-class scatter of the projected samples.

The objective now is to maximize the energy function in (2.9). In other words, we would like to obtain a projection where samples of the same class are as close as possible to each other while as further apart as possible from samples of the other class. To find the optimal transformation that maximizes the energy function, we need to rewrite equation (2.9) as a function of $\boldsymbol{a}$. Substituting equation (2.10) into the numerator of the energy function results in

$$\begin{aligned}\left(\widetilde{\mu}_1 - \widetilde{\mu}_2\right)^2 &= \left(\boldsymbol{a}^T\boldsymbol{\mu}_1 - \boldsymbol{a}^T\boldsymbol{\mu}_2\right)^2 = \boldsymbol{a}^T\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^T\boldsymbol{a} \\ &= \boldsymbol{a}^T\boldsymbol{S}_B\boldsymbol{a} = \widetilde{\boldsymbol{S}}_B\end{aligned} \tag{2.12}$$

where $\boldsymbol{S}_B$ is a matrix and is called the between-class scatter of the samples in the original feature space and $\widetilde{\boldsymbol{S}}_B$ is the projected samples' between-class scatter matrix.

To have the within-class scatter matrix of the projected samples in terms of $\boldsymbol{a}$ as well, we can use equation (2.11) as follows

$$\begin{aligned}\widetilde{s_i}^2 = \sum_{y \in w_i} (y - \widetilde{\mu}_i)^2 = \sum_{\boldsymbol{x} \in w_i} \left(\boldsymbol{a}^T \boldsymbol{x} - \boldsymbol{a}^T \boldsymbol{\mu}_i\right)^2 = \sum_{\boldsymbol{x} \in w_i} \boldsymbol{a}^T (\boldsymbol{x} - \boldsymbol{\mu}_i) (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \boldsymbol{a} \\ = \boldsymbol{a}^T \boldsymbol{S}_i \boldsymbol{a}\end{aligned} \tag{2.13}$$

where $\boldsymbol{S}_i$ is a measure of the variation of original samples in class w_i. We can rewrite the denominator of the energy function in (2.9) (which is in fact the within-class scatter of the projected samples) as follows

$$\widetilde{s_1}^2 + \widetilde{s_2}^2 = \boldsymbol{a}^T \boldsymbol{S}_1 \boldsymbol{a} + \boldsymbol{a}^T \boldsymbol{S}_2 \boldsymbol{a} = \boldsymbol{a}^T (\boldsymbol{S}_1 + \boldsymbol{S}_2) \boldsymbol{a} = \boldsymbol{a}^T \boldsymbol{S_W} \boldsymbol{a} = \widetilde{S}_W \tag{2.14}$$

Finally, equation (2.9) can be reformulated as a function of the transformation vector $\boldsymbol{a}$, $\boldsymbol{S}_B$, and $\boldsymbol{S}_W$:

$$J(\boldsymbol{a}) = \frac{|\widetilde{\mu}_1 - \widetilde{\mu}_2|^2}{\widetilde{s_1}^2 + \widetilde{s_2}^2} = \frac{\boldsymbol{a}^T \boldsymbol{S}_B \boldsymbol{a}}{\boldsymbol{a}^T \boldsymbol{S}_W \boldsymbol{a}} \tag{2.15}$$

The transformation vector that maximizes $J(\boldsymbol{a})$, $\boldsymbol{a}^*$ is called Fisher's linear discriminant and it turns out that it is in fact the eigenvector of the following matrix [63]:

$$\boldsymbol{S}_X = \boldsymbol{S}_W^{-1} \boldsymbol{S}_B \tag{2.16}$$

2.3.2.2 C-classes case

In the case of having C classes, the maximum dimension of the projected samples can be $C - 1$. So in this case, our objective will be to find the $C - 1$ projection vectors that result in the maximum between class and minimum within-class variances. Let's populate these projection vectors in a matrix of size $p \times (C - 1)$ called $\boldsymbol{A}$. We use equation (2.17) to project the entire samples in the original p-dimensional space to the

new $(C-1)$-dimensional feature space.

$$\boldsymbol{Y} = \boldsymbol{A}^T \boldsymbol{X} \tag{2.17}$$

where $\boldsymbol{X}$ is a $p \times N$ matrix including the original dataset and $\boldsymbol{Y}$ is a matrix of size $(C-1) \times N$ consisting of the projected samples in the lower dimensional space.

To find the within-class variance for the C classes case, similar to the previous case we need to add the scatter matrices of all the C classes:

$$\boldsymbol{S}_W = \sum_{i=1}^{C} \boldsymbol{S}_i = \sum_{i=1}^{C} \sum_{\boldsymbol{x} \in w_i} (\boldsymbol{x} - \boldsymbol{\mu}_i) (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \tag{2.18}$$

where $\boldsymbol{\mu}_i$ is the mean of the original samples in class w_i.

To find the between-class scatter of original samples, $\boldsymbol{S}_B$, we use the mean of all samples in the dataset, $\boldsymbol{\mu}$ according to (2.19):

$$\boldsymbol{S}_B = \sum_{i=1}^{C} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu}) (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{2.19}$$

Similarly, the within and between- class scatter matrices for the projected samples ($\widetilde{\boldsymbol{S}}_W$ and $\widetilde{\boldsymbol{S}}_B$, respectively) can be expressed in equations (2.20) and (2.21), receptively.

$$\widetilde{\boldsymbol{S}}_W = \sum_{i=1}^{C} \widetilde{\boldsymbol{S}}_i = \sum_{i=1}^{C} \sum_{\boldsymbol{y} \in w_i} (\boldsymbol{y} - \widetilde{\boldsymbol{\mu}}_i) (\boldsymbol{y} - \widetilde{\boldsymbol{\mu}}_i)^T \tag{2.20}$$

$$\widetilde{\boldsymbol{S}}_B = \sum_{i=1}^{C} N_i (\widetilde{\boldsymbol{\mu}}_i - \widetilde{\boldsymbol{\mu}}) (\widetilde{\boldsymbol{\mu}}_i - \widetilde{\boldsymbol{\mu}})^T \tag{2.21}$$

where $\widetilde{\boldsymbol{\mu}}_i$ and $\widetilde{\boldsymbol{\mu}}$ are the mean of projected samples of class w_i and the mean of all projected samples, respectively.

It should be noted that equations (2.12) and (2.14) hold for the C-classes case as well. Since in the case of having more than two classes, $\widetilde{\boldsymbol{S}}_W$ and $\widetilde{\boldsymbol{S}}_B$ are matrices rather than scalar values, we use their determinant value in the equation of the energy function.

$$J(A) = \frac{\left|\widetilde{\boldsymbol{S}}_B\right|}{\left|\widetilde{\boldsymbol{S}}_W\right|} = \frac{\left|\boldsymbol{A}^T \boldsymbol{S}_B \boldsymbol{A}\right|}{\left|\boldsymbol{A}^T \boldsymbol{S}_W \boldsymbol{A}\right|} \tag{2.22}$$

The desired matrix, $\boldsymbol{A}^*$, is the one that maximizes the energy function in (2.22). It can be mathematically proven that similar to the two-classes case, such $\boldsymbol{A}^*$ matrix is composed of the eigenvectors of the following matrix

$$\boldsymbol{S}_X = \boldsymbol{S}_W^{-1} \boldsymbol{S}_B. \tag{2.23}$$

$\boldsymbol{A}^*$ is a $p \times (C-1)$ matrix whose columns are the eigenvectors of matrix $\boldsymbol{S}_X$ in equation (2.23) and are located in an descending order such that the jth column corresponds to the jth largest eigenvalue of $\boldsymbol{S}_X$.

2.3.3 Artificial neural networks (ANN)

Artificial neural networks are machine learning based classifiers whose architecture is inspired by the way human brain works. An ANN is composed of connected units called artificial neurons which are similar (but not the same) as human biological brain neurons. Each neuron has some main components: cell body, dendrites, and axon. Cell body is the main computational unit of a neuron. Dendrites act as input wires which receive input data from other neurons. Each neuron may own thousands of these input receivers which are usually short in length. Axons, as opposed the dendrites, are the neuron's output wires which send information signals to other neurons and can break into thousands of branches at the end called axon terminals. Axons own a single long shape which can be up to one meter in length.

2.3.3.1 Single layer neural network (perceptron)

In an ANN, instead of a neural signal, what each neuron receives is a sum of weighted numbers/inputs. This sum will then go through a non-linear function to provide the neuron's output. Figure 2.5 shows a graphical representation of a single layer neural network including only one hidden neuron. In fact, such a network performs like a perceptron, a binary linear classifier. The output of a single layer NN can be formulated as (2.24):

$$y = f\left(\sum_{j=1}^{p} w_j x_j + b\right) \tag{2.24}$$

where x_j is the jth element of the input vector $\boldsymbol{x}$ and w_j is the corresponding weight between that input and the hidden neuron. Also, b represents a bias value. f is the neuron's non-linear activation function.

Activation functions introduce nonlinear properties to our model and let the network learn complex mappings from the input to the output. In fact, we can consider ANN as *universal function approximator* which is capable of learning any function provided that it contains nonlinear activation function. A neural network without activation function resembles a linear regression model which is not capable of modeling real complex nonlinear relationships between the input and output of the network, so, fails to perform properly in real applications. Besides the nonlinear property, an activation function must be differentiable. The need for this property will be explained in Section 2.3.3.2. Figure 2.6 shows the most commonly used activation functions: sigmoid or logistic, tangent hyperbolic (*tanh*), and rectified linear unit (ReLU). What follows is a brief description of these functions.

Sigmoid or logistic activation function: This function is defined by (2.25)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.25}$$

where z is the function's input. So, the output of a neuron with input $\boldsymbol{X} = [x_1, x_2, ..., x_p]$,

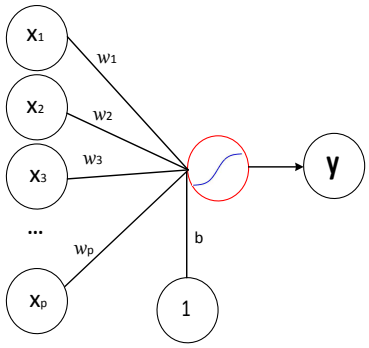
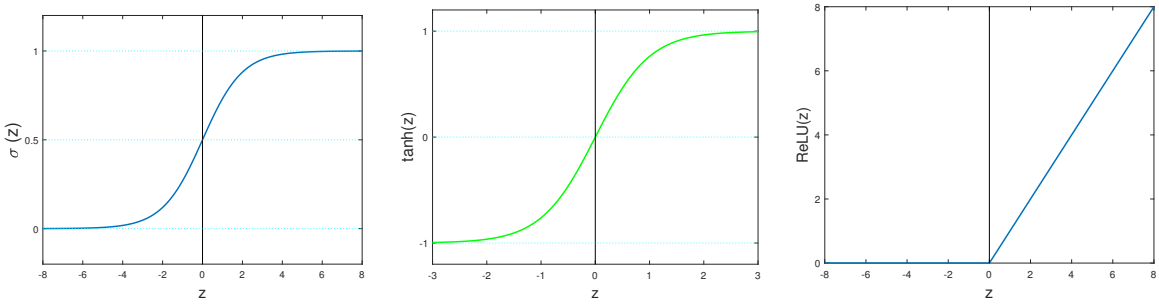


Figure 2.5: Graphical representation of a single layer NN with only one hidden neuron which functions similar to a perceptron.



(a) (b) (c)

Figure 2.6: Several activation functions. (a) Sigmoid, (b) tanh, (c) ReLU.

the corresponding weights $\boldsymbol{W} = [w_1, w_2, ..., w_p]$, bias value b, and the sigmoid activation function can be represented as (2.26)

$$\sigma(z) = \frac{1}{1 + e^{-\left(\sum_{i=1}^{p} x_i w_i + b\right)}}. \tag{2.26}$$

As can be seen from Figure 2.6a the output of the sigmoid function is bounded between 0 and 1 which makes it especially useful when we expect our model to predict the probability of a class (or event) to happen. This function is differentiable at all points which is one of the reasons behind its popularity in ANN applications. However, the drawback of using this sigmoid function is that since for the majority of the x domain its derivative has a very small value close to zero it can lead to the *vanishing gradient* dilemma [65].

Tanh activation function: This function is the ratio between the hyperbolic sine and the cosine functions and is defined as

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.27}$$

where z is the weighted sum of inputs to the neuron. The shape of this function is quite similar to the sigmoid function (see Figure 2.6b) but it is bounded between -1 and 1. Similar to the sigmoid function, tanh is differentiable at all points.

ReLU activation function: This activation function has become popular recently especially in CNN applications. Considering z as the input and $f(z)$ as the output of this function, the relationship between these two can be formulated as (2.28)

$$f(z) = \begin{cases} z & \text{if} \quad z \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.28}$$

As can be seen from Figure (2.6c), it is differentiable at all points. Also, since the gradient of this function is equal to one for the positive inputs, it solves the vanishing gradient problem making it a popular activation function. Nonetheless, ReLU has some drawbacks as well. First, it should only be used in hidden layers of a NN and for a classification problem, in the output layer, *Softmax* function must be employed. Second, using the ReLU function, all the negative values as inputs of the function will be mapped to zero immediately which decreases the ability of the model to fit or train from the data properly.

2.3.3.2 Multi layer neural network

To obtain a more powerful classifier than a perceptron with more complex decision boundaries, multiple perceptrons are combined and function together. They are arranged in layers such that each neuron takes the sum of its weighted inputs, apply the activation

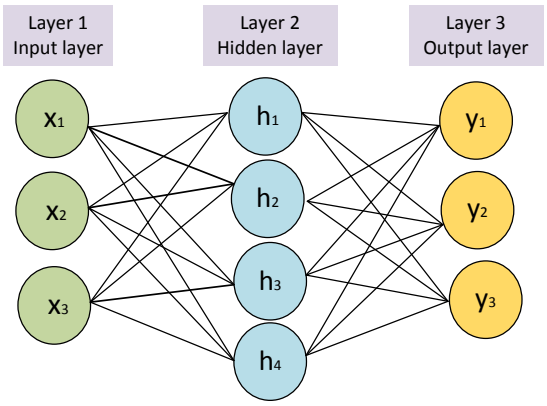


Figure 2.7: Schematic of a sample multilayer neural network with input data with three features, one hidden layer with four hidden units, and the output layer with three nodes corresponding to a three-class classification problem.

function on it, then transmits the output to the neurons in the next layer. Such an architecture is called multilayer neural network. Schematic of a simple multilayer NN is shown in 2.7. The first layer in the network is called *input layer* and accepts the input data. The middle layer called *hidden layer* is composed of hidden *units* or hidden *nodes* and the third layer is called *output layer* whose number of nodes for a classification problem is equal to the number of classes in the dataset. It should be noted that the network in Figure 2.7 is only an example and in general, according to the specifications of the problem at hand, a multilayer NN can have as many input, hidden, and output nodes and also hidden layers as needed. For the general case of having inputs with p features, n hidden units, L hidden layers and c classes, the following equations can be established:

$$h_j^l = f_1\left(\sum_{i=1}^{p} w_{ij}^l x_j + b_j^l\right) \qquad \text{for} \qquad l = 1, ..., L \quad \text{and} \quad j = 1, ..., n \tag{2.29}$$

$$y_k = f_2\left(\sum_{j=1}^{n} w_{jk}^o h_j^L\right) \qquad \text{for} \quad k = 1, ..., c \tag{2.30}$$

where w_{ij}^l and w_{jk}^o refer to the weights of the lth hidden layer and the output layer of the network. f_1 and f_2 are the activation functions of the hidden layer(s) and the output layer, respectively (sometimes the same activation function is used for both).

Like any other classifier, ANN needs to be trained by the means of training examples and updates its parameters. These trainable parameters in an ANN include the connections' weights and biases of different layers of the network. During the training stage of an NN, training examples are given to the network and network assigns a label to each one. The discrepancy between the desired output and the network output is in fact a function of the weights of the network and is called *loss function.* The objective of training a NN is to find weights and biases which minimize this loss function. Gradient descent (GD) [65] or its variations like stochastic gradient descent (SGD) are the most commonly used optimization algorithms to achieve this goal. To find the optimal values for the network's parameters, GD starts by initializing the weights randomly. Then at each time step t, takes a step (by updating the weights and biases) on the loss function surface in the multi-dimensional space in the direction of decreasing gradient aiming at gradually reaching the global minimum of the loss. It can be mathematically proven that the partial derivative of the loss function with respect to the weights of the network is a function of the neurons' activation function. Therefore, in order to be able to compute these partial derivatives, the activation function needs to be differentiable. The equations for updating the weights and biases of a network using GD are given by (2.31) and (2.32)

$$w_k^{t+1} = w_k^t - \alpha \frac{\partial L}{\partial w_k} \tag{2.31}$$

$$b_k^{t+1} = b_k^t - \alpha \frac{\partial L}{\partial b_k} \tag{2.32}$$

where w_k^{t+1} (b_k^{t+1}) and w_k^t (b_k^t) are the value of the kth weight (bias) of the network at time steps $t+1$ and t, respectively and $\partial(.)$ represents the partial derivative. Also, α is a preset value and is called *learning rate* which is related to the speed of the convergence of the optimization process.

In order to compute the gradient of the loss function with respect to the different parameters of the network, an algorithm called *backpropagation* (BP) [66] is used. This method is widely used in supervised machine learning, specifically deep learning for training neural networks. The term BP and its application in neural networks was

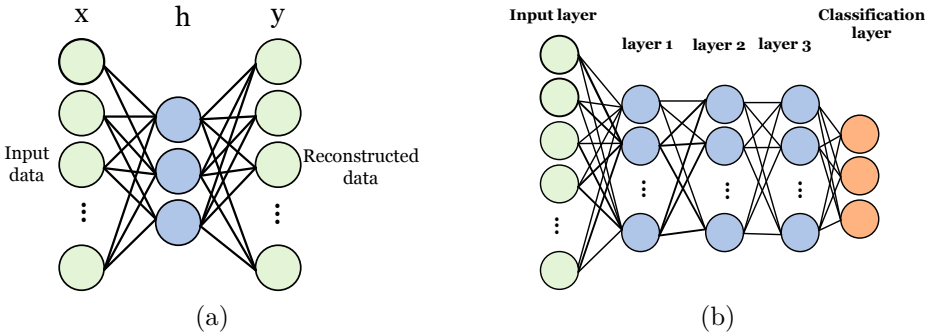


Figure 2.8: Block diagram of (a) a sample auto-encoder and (b) stacked autoencoder

introduced by Rumelhart *et al* in 1986 [67] where it was used in multiple neural networks showing its faster performance compared to its preceding learning approaches. For a comprehensive explanation of this algorithm please see Chapter 2 of [65].

2.3.4 Deep neural network

A deep learning model, also called deep neural network, is composed of a neural network with typically more than three layers. In a deep learning model abstract features which are associated with higher layers are learned from simple, low level features from previous layers in the network. In fact, the goal of all deep learning methods is to learn representative, abstract, and discriminative features from the input data, automatically. Deep learning has been successfully employed in many areas such as speech recognition [68] and face recognition [69, 70]. A deep network can be trained either in a supervised or an unsupervised manner. Common deep network architectures include stacked autoencoder, deep belief network, recurrent neural network (RNN), and convolutional neural network.

2.3.4.1 Stacked autoencoder

Figure2.8b shows a sample SAE network. SAE is composed of layers of autoencoders. An autoencoder (AE) is composed of an input, hidden, and output layers with dimensions d, h, and d, respectively where usually $h < d$. There are weighted connections between units in the input and hidden layers as well as hidden and the output layers. Figure 2.8a

shows a typical AE. Training an AE consists of two steps: encoding and reconstruction. During the encoding phase, data in the input layer is mapped (encoded) to a new feature space (hidden layer). Reconstruction phase (decoding) aims to reconstruct the original data from the mapped features. These two steps can be written as (2.33) and (2.34), respectively.

$$\boldsymbol{h} = f(\boldsymbol{W}_h \boldsymbol{x} + \boldsymbol{b}_h) \tag{2.33}$$

$$\boldsymbol{y} = g(\boldsymbol{W}_y \boldsymbol{h} + \boldsymbol{b}_y) \tag{2.34}$$

where $\boldsymbol{W}_h$ is the weights of the connections between input units to the hidden units and $\boldsymbol{W}_y$ is the weights of the connections between the hidden units and the output units. $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ stand for the biases of the hidden and output units, respectively. Also, f and g are the activation functions of the neurons in the hidden and output layers, respectively (for more information about the activation functions see Section 2.3.3.1) and in practice, in most of the cases these two functions are set to be the same. Training an AE aims to find features that represent the input data in the best way such that the reconstruction error is minimized. In fact, these features are obtained during an optimization process which minimizes a cost function representing the difference between the output (which is a reconstructed version of the input data) and the input data itself according to (2.35):

$$\boldsymbol{\theta} = \arg \min_{\mathbf{W},\boldsymbol{b}} \sum_{i=1}^{M} C\left(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\right) \tag{2.35}$$

where $C(\boldsymbol{.})$ is the cost function. M and $\boldsymbol{\theta}$ are the total number of training examples and the best set of parameters found during the optimization process, respectively.

There are some different variations for AEs including sparse and denoising AE [66]. Since in this thesis we employed sparse AE in our deep learning framework, the following subsection presents an introduction to this model.

2.3.4.2 Sparse autoencoder

A sparse AE is a model whose training loss function involves a sparsity regularization term. This regularization term imposes the sparsity constraint on the output of the AE's hidden layer and is a function of the mean of the output value of the hidden layer's neurons [71]. This mean value can be represented as (2.36):

$$\widehat{\rho}_j = \frac{1}{M} \sum_{i=1}^{M} f\left(\boldsymbol{W}_j \boldsymbol{x}_i + b_j\right) \tag{2.36}$$

where $\boldsymbol{W}_j$ and b_j are the vector of weights and the bias value of the jth neuron, respectively. A neuron is activated if its activation value is high enough. Having a low value for $\widehat{\rho}_j$ means that neuron j responses only to a small subset of training data having distinct features. Adding a term to the loss function that measures the difference between the $\widehat{\rho}_j$ and the desired neuron's output value, called sparsity regularization, makes each neuron master learning features belonging to only a small portion of training samples. Kullback-Leibler divergence [72] can be considered as such sparsity regularization term:

$$\begin{aligned} &\gamma_{sparsity} = \sum_{j=1}^{N} KL\left(\rho || \widehat{\rho}_j\right) = \\ &\sum_{j=1}^{N} \rho \log\left(\frac{\rho}{\widehat{\rho}_j}\right) + (1-\rho) \log\left(\frac{1-\rho}{1-\widehat{\rho}_j}\right) \end{aligned} \tag{2.37}$$

where N and ρ are the number of neurons in the hidden layer and the desired value for the average activation of each neuron, respectively.

One drawback of having such a sparsity regularization term in the cost function is that during the training process, high values may be assigned to the weights of the network in order to maximize $\widehat{\rho}_j$. To prevent this case to happen, a weight regularization term should be added to the AE's cost function as shown in (2.38)

$$\gamma_{weights} = \frac{1}{2}\sum \|\boldsymbol{W}\|_2^2 \tag{2.38}$$

where $\|\boldsymbol{W}\|_2$ is the L_2 norm of the AE's weight matrix. Finally, the cost function of the sparse AE can be expressed as (2.39):

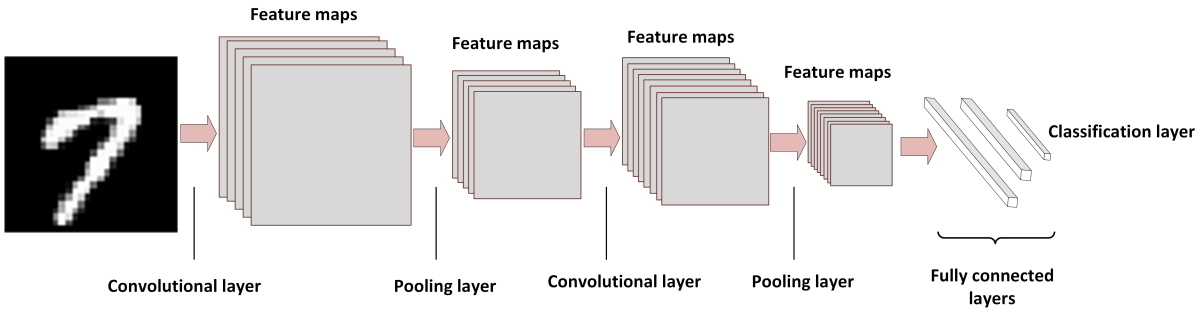
$$J = \frac{1}{M}\sum_{i=1}^{M}(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^2 + (\lambda \times \gamma_{weights}) + (\beta \times \gamma_{sparsity}) \tag{2.39}$$

where λ and β are the L_2weight regularization and sparsity regularization coefficients, respectively.

Arranging AEs one after another such that the hidden layer of one AE becomes the input layer of the next, constructs a SAE. This arrangement aims to extract deep features at higher layers. Training a SAE consists of two phases: pre-training and fine-tuning.

In the pre-training phase, each AE is trained in an unsupervised manner. In other words, labels of the input samples are not used during pre-training. For each AE, having found the weights and biases that minimize the cost function, reconstruction layer together with its weights and biases are removed. Hidden layer's features are stored and given to the next AE as input while its weights and biases are saved to be used in the next step.

In the fine-tuning phase, all layers are connected forming a unified network and the input of the network will be the original input data as well as their labels. Also, putting a classification layer on top the network provides the supervised training. The important characteristic of this phase is that parameters of the network are not initialized randomly at the beginning of the optimization process, rather this initialization is done using the weights and biases which have been already obtained and saved in the pre-training phase. In fact, after adding the classification layer and training the whole network to find the final optimal weights and biases, they only slightly change about the values already obtained in the pre-training stage [73]. In other words, compared to the extracted deep features in the pre-training step, class labels provide limited information which is used



(a)

k_1	k_2	k_3
k_4	k_5	k_6
k_7	k_8	k_9

(b)

Figure 2.9: (a) A sample convolutional neural network with two convolution, two pooling, and three fully connected layers (b) convolution kernel.

only for marginally adjusting the weights and biases of the network about their initial values [73].

2.3.4.3 Convolutional neural network

The first research on modern CNN was carried out by LeCun *et al.,* in 1998 [74] to perform handwritten character recognition. The special architecture of CNNs make them well-suited for image classification. The reason behind the name "convolutional" is the usage of the linear mathematical operation called "convolution" by the network. To put it simply, CNNs are neural networks that replace matrix multiplication with convolution at least in one of their layers [66]. Figure 2.9a shows a sample convolutional neural network. As can be seen from this figure, a CNN is typically composed of convolutional, pooling, and fully connected (FC) layers where the FC layers are usually placed on the top of the network. The feature maps in this figure are either results of the convolution or the pooling operations. In the convolution, the input image (or previous feature maps) are convolved with a kernel (filter) of size $n \times n$. Figure 2.9b shows a sample convolution kernel of size 3×3. In practice, we use multiple of these kernels in each of

convolution layers each producing a separate feature map. In CNNs, a pooling layer is usually placed right after a convolution layer to simplify the information in the output from the convolutional layer. The most common type of pooling in CNNs is max-pooling where in the small window (e.g. 2×2) in different locations of the feature map the maximum value is found and replaces the whole window. Two important characteristics of CNNs are local connectivity and shared weights.

2.3.4.4 Deep belief network

A DBN is another type of DNN and is constructed of layers of RBM. An RBM is a two layer network composing of visible (v) and hidden (h) layers. Diagram of a sample RBM is shown in Figure 2.10. The energy of the joint configuration of the units is defined as (2.40)

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{a}^T\boldsymbol{h} - \boldsymbol{b}^T\boldsymbol{v} - \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h} \tag{2.40}$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ are the visible and hidden vectors which we would like to compute the energy for, $\boldsymbol{a}$ and $\boldsymbol{b}$ are the biases of the hidden and visible layers, respectively, and $\boldsymbol{W}$ is the matrix of the weights between the two layers of the network. The joint probability distribution of the visible and hidden vectors is a function of the energy of their configuration defined in (2.40) and is defined as

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{2.41}$$

where Z is the normalizing constant and is defined as

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{2.42}$$

The probability that the network assigns to a visible vector, $\boldsymbol{v}$, is given by summing

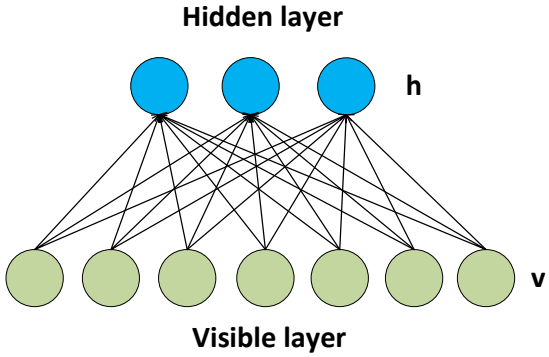


Figure 2.10: A sample RBM.

over all possible hidden vectors:

$$p(\boldsymbol{v}) = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{2.43}$$

The conditional distributions of hidden unit h_j is given by logistic function

$$p(h_j = 1|\boldsymbol{v}) = \sigma(\sum_{i=1}^{D} w_{i,j}v_i + a_j) \tag{2.44}$$

where D is the dimension of the visible vector and σ represents the sigmoid function defined in (2.25). What (2.44) means is in fact the probability of the hidden unit h_j to be set to 1 provided that the given visible vector is $\boldsymbol{v}$. We can have similar equation for the conditional distributions of the visible units. However, since most of the time visible (input) vectors are real valued data (not binary), we assume that the visible units, conditioned on the hidden layer, are independent gaussian random variables defined as

$$p(v_i|\boldsymbol{h}) = \mathrm{N}(\sum_{j=1}^{F} w_{i,j}h_j + b_i) \tag{2.45}$$

where F is the size of the hidden layer and $N(.)$ stands for the gaussian function.

The goal of training each RBM is to maximize the likelihood of all training examples and as suggested in [75], parameters of the network can be adjusted by applying a stochastic gradient descent on the log likelihood of all training samples. In other words,

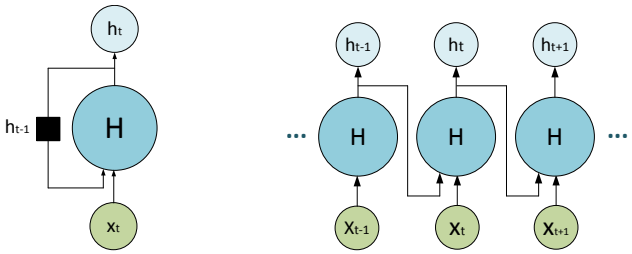


Figure 2.11: (a) Schematic of an RNN (b) unfolded network shown in (a).

the optimal weights and biases of an RBM can be calculated as

$$\boldsymbol{\theta} = \arg \min_{\boldsymbol{W},\boldsymbol{b}} \left(-\sum_{m} \log(p(\boldsymbol{v}_m)) \right) = \arg \min_{\boldsymbol{W},\boldsymbol{b}} \left(-\sum_{m} \log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v_m},\boldsymbol{h}_m)} \right). \quad (2.46)$$

Having obtained the optimal parameters for each RBM, several of these machines can be stacked together and form a DBN such that the hidden layer of one RBM becomes the visible layer of the next. Similar to SAE, training process of a DBN consists of the two steps of pre-training and fine-tuning. Therefore, the optimal parameters of each individual RBM is used as the initial values for the fine-tuning step.

2.3.4.5 Recurrent neural network

RNN is another type of deep neural networks which is used for processing dynamic data such as video or sequences of text. The main difference between RNN and a conventional feed forward neural network is that in an RNN, the output of each hidden neuron is fed back to itself as input introducing the concept of time to the network. Figure 2.11 is a schematic of an RNN. The black square in the left subfigure indicates the time delay by one step and means that the output of each hidden neuron at time step t is given back to the neuron as an extra input at time step $t+1$. Figure 2.11b shows an unfolded version of the network shown in Figure 2.11a. As can be seen from this figure at each time step t, a combination of the input at that step $\boldsymbol{x}_t$ and the output of the previous time step $\boldsymbol{h}_{t-1}$ is given to the hidden neuron as input. Like any other type of NN, RNN

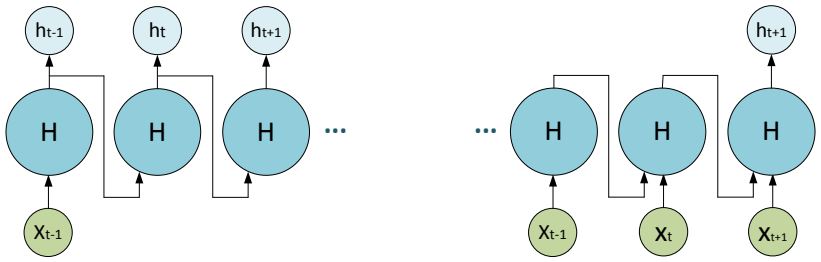


Figure 2.12: RNN (a) One to many and (b) many to one architectures.

also includes weights and biases. But, the important thing is that these parameters are shared with different time steps in the network. In other words, there is not a different set of parameters for each time step. Outputs at each time step t are produced by the means of the same parameters' update rule [66].

RNNs own flexible architecture. In fact the one shown in Figure 2.11b is only one of them called *many to many* model where many inputs $\{..., \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, ...\}$ are mapped to many outputs $\{..., \boldsymbol{h}_{t-1}, \boldsymbol{h}_t, \boldsymbol{h}_{t+1}, ...\}$. Two other architectures include *one to many* and *many to one* models which are shown in Figures 2.12a and 2.12b, respectively.

The common problem with vanilla RNN is the vanishing (or rarely exploding) gradient problem. Ideally, we would like to figure out the relationship of data at many time steps even if there are far from each other. However, The more time steps in the network means the more gradient multiplications during the backpropagation process. Considering the sigmoid function as the neurons' activation function whose gradient is small (close to zero) for the majority of the inputs, multiplying such small values (whose number increase with the number of time steps in our RNN) will result in very small amount of update for weights during the training time. So, RNNs in their basic format do not perform well in many real applications. The most effective approach to reduce the vanishing gradient problem in RNNs so far is using long short term memory (LSTM) cells. To read more about the architecture of the LSTM cell please read [66].

Since RNN is used for processing sequential data where there is different input/output at each time step, it is not suitable for our purpose since in this project, we deal with

processing static hyperspectral datasets.

2.3.5 Extended multi-attribute profile

Attribute profiles are obtained by applying morphological attribute filters on an image. Compared to conventional morphological profiles [76], they extract more powerful spatial features representing important spatial information from the input image. Morphological attribute filters allow us to process the input images with less computational cost and extract various types of features corresponding to both scale and texture. Moreover, the definition of the attributes are more flexible in this method compared to MP where the shape of the SE is fixed. To have a better understanding of APs let's first describe MPs.

2.3.5.1 Morphological profiles

Morphological profiles are obtained by applying morphological operators of opening and closing by reconstruction on an input image [33]. The definitions of opening and closing by reconstruction are given by (2.47) and (2.48), respectively.

$$\gamma_R^i(I) = R_I^\delta(\varepsilon^i(I)) \tag{2.47}$$

$$\varphi_R^i(I) = R_I^\varepsilon(\delta^i(I)) \tag{2.48}$$

where I is an input image, γ_R and φ_R are opening and closing by reconstruction, δ^i and ε^i represent dilation and erosion operations with a SE of size i, and R_I^δ and R_I^ε are the geodesic reconstruction by dilation and erosion, respectively.

We can obtain opening and closing profiles by applying (2.47) and (2.48), respectively on an input image with SE of fixed shape and increasing size. Finally, the MP is obtained

by stacking closing and opening profiles according to (2.49) [36]:

$$\mathrm{MP}(I) = \mathrm{II}_i : \begin{cases} \mathrm{II}_i = \mathrm{II}_{\gamma\lambda} & \text{with } \lambda = (i-n-1), \quad \forall i \in [n+1, 2n+1] \\ \mathrm{II}_i = \mathrm{II}_{\varphi\lambda} & \text{with } \lambda = (n-i+1), \quad \forall i \in [1, n] \end{cases} \tag{2.49}$$

where $\mathrm{II}_{\gamma\lambda}$ is the concatenation of opening profile obtained by the SEs of increasing size (from 0 to n, where opening profile with SE of size zero corresponds to the original image) and $\mathrm{II}_{\varphi\lambda}$ represents the closing profile obtained by the SEs of decreasing size (from n to 1). Finally, $\mathrm{MP}(I)$ is the concatenation of these two profiles.

2.3.5.2 Extended morphological profiles (EMP)

MP with its definition in (2.49) can only be applied on one-band images. Therefore, in order to be able to extract MPs of multi-band images (e.g., multi spectral or hyperspectral), this method needed to be further explored. In [34], Benediktsson *et al* performed some modifications on the MP algorithm [33], and successfully applied it on hyperspectral datasets. The new algorithm called extended morphological profile (EMP) is explained below.

First PCA is applied on the hyperspectral image along its spectral dimension and the first few PCs including the most variation in the dataset is kept. Then, MP of each retained PC image is computed according to (2.49). Finally, the EMP of the input HSI, is obtained by stacking the MPs of the preserved PCs according to (2.50)

$$\mathrm{EMP}(I) = \{\mathrm{MP}(\mathrm{PC}_1(I)), \mathrm{MP}(\mathrm{PC}_2(I)), ..., \mathrm{MP}(\mathrm{PC}_k(I))\} \tag{2.50}$$

where I and $\mathrm{PC}_i(I)$ are the input HSI and the ith retained PC of it, respectively.

EMP has been successfully applied to extract spatial information of images. However, there are some problems associated with it: 1. High computational complexity. 2. only analysis of the scale is included in the processing. 3. The limitation of the extracted

spatial features due to the usage of fixed- shaped SE. Attribute profiles as a solution for these drawbacks have been proposed in 2010 by Dalla Mura *et al* [36]. Attribute profiles are obtained by applying morphological attribute filters on an image. Next section presents a description of these filters and how they are employed to build attribute profiles.

2.3.5.3 Attribute profiles

Various structural information in an image can be modeled by subsequently applying morphological attribute filters on the image leading to morphological attribute profiles. Attribute filters are in fact morphological attribute opening and attribute thinning developed by Breen *et al* [77] in 1996. An attribute can be considered as any measure which can be computed on the connected components of an image. Examples of such attributes include area and length of the perimeter of the connected component (size-related attributes), standard deviation of the pixels in each connected component (textural attribute), length of the diagonal of the bounding box of the connected component (shape-related attribute), etc. Attribute filters are connected component transformations and since are quite similar to the morphological opening and closing by reconstruction. A common property of operators by reconstruction and attribute filters is that they do not produce new edges to the image since they either entirely remove or keep a structure from the input image. However, they perform differently considering the fact that the former transforms the input image based on the size of a fixed-shape SE while the latter performs transformation based on different attributes computed on the connected components which do not necessarily reveal information about the size of the structures. To understand how these attribute operators work let's first consider the case when the input is a binary image.

Binary attribute filters: The first step to apply these operators on a binary image is to detect connected components (CCs) in the image. Then, for each CC, different criteria are evaluated and if it meets the criterion T, the CC is preserved otherwise is

removed from the image (i.e. its pixels are set to zero). These criteria may be increasing such as area, size of the bounding box, volume[1], etc. or non-increasing like of the shape factors[2]. If the former, we call the operation attribute opening, if the latter we call it attribute thinning [77]. Binary attribute opening on image I can be defined by (2.51):

$$\Gamma^T(I) = \bigcup_{x \in I} \Gamma_T(\Gamma_x(I)) \tag{2.51}$$

where x is any pixel in I and $\Gamma_x(I)$ is called *binary connected opening* and returns the CC including pixel x. $\Gamma_T(.)$ is called *binary trivial opening* and is defined by (2.52)

$$\Gamma_T(C) = \begin{cases} C, & \text{if C meets the criterion T} \\ 0, & \text{otherwise.} \end{cases} \tag{2.52}$$

where C is a connected component. Finally, the binary attribute opening of image I ($\Gamma^T(I)$) is obtained by the union of the results of applying binary trivial opening with the criterion T on each connected component of the I.

Analogously, binary attribute thinning can be defined by (2.53):

$$\widehat{\Gamma}^T(I) = \bigcup_{x \in I} \widehat{\Gamma}_T(\Gamma_x(I)). \tag{2.53}$$

The above definitions and equations can be extended to define the dual transformations *binary attribute closing* $\varphi^T(I)$ and *binary attribute thickening* $\widehat{\varphi}^T(I)$.

Grayscale attribute filters: Binary attribute operators defined for binary images can be generalized to the grayscale case. One of the approaches to do this generalization is by using *threshold decomposition* method [76]. In this method, K binary images are produced from the gray image I where K is the number gray levels presented in the

[1]Volume of a CC is the sum of the gray level values of all pixels inside the connected component.

[2]Shape factors of a region is independent of its size and are computed from some features such as the diameter, area, and perimeter of the region.

image. Each of these binary images can be defined by (2.54)

$$I_i(x) = \begin{cases} 1, & \text{if} \quad I(x) \geq i \\ 0, & \text{otherwise} \end{cases} \tag{2.54}$$

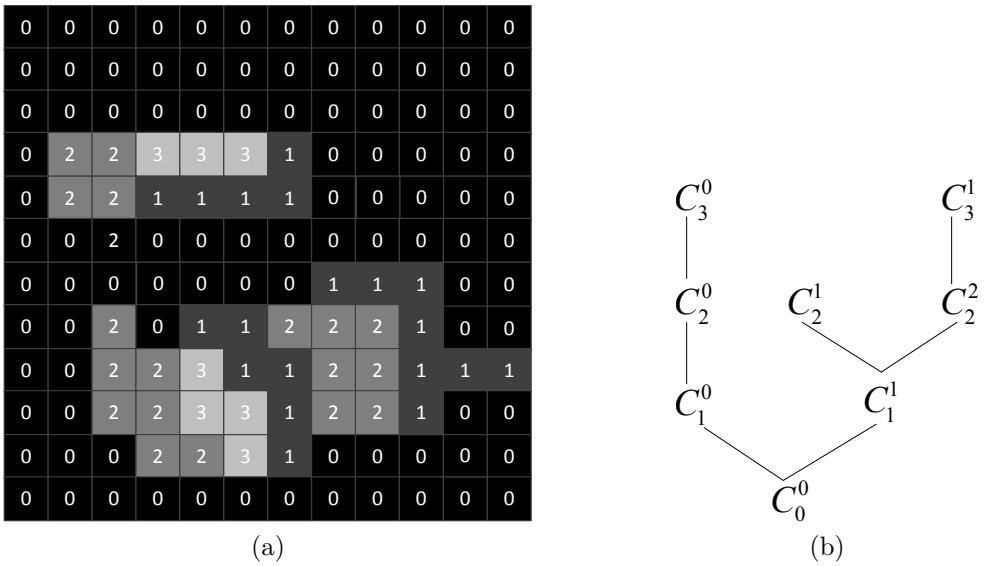
where x is any pixel in image I and i is one of the K gray levels existing in image I. Having decomposed the gray image to a stack of binary images, binary attribute opening (thinning) is applied on each one. Finally, the value of each pixel at the output image will be the maximum graylevel of the results of the filtering for each pixel.

Although the approach that has been described is doable, it is not computationally very efficient. Instead one can use the *max-tree* representation of a grayscale image introduced by Salembier *et al.* [78] to efficiently perform the attribute opening/thinning filtering on grayscale images. The next section explains the structure of a Max-tree and how it is used in performing attribute opening/thinning.

2.3.5.4 Max-Tree

Performing attribute filtering on grayscale images using the max-tree representation, includes the following steps:

- **Max-tree creation:** In this step, the image is represented in a tree structure with different levels where each level corresponds to a gray level present in the image and the number of nodes in each level is equal to the number of connected components at that gray level. The root of the tree is associated with the lowest gray level in the image. Also, a node at gray level $i+1$, n_{i+1}, is connected to a node at gray level i, n_i, if n_i corresponds to a connected component that contains the one associated with n_{i+1}. In this case, n_i and n_{i+1} are called parent and child nodes, respectively. A synthetic gray-level image and its corresponding max-tree structure is shown in Figure 2.13.



(a) (b)

Figure 2.13: Max -tree representation. (a) a synthetic gray-scale image and (b) max-tree structure of image in (a).

- **Attribute calculation:** At this step, for each node in the tree which is in fact a connected component, the value of attribute A is computed (e.g. the area of the connected component).

- **Filtering:** At the filtering step, a criterion T is evaluated for each node. In other words, at each node the value of attribute A is evaluated with respect to a threshold value and the tree is pruned by removing the nodes that do not meet the evaluated criterion.

- **Image restitution:** Finally, to have the result of applying attribute filter in the form of a gray-scale image again, the pruned tree structure is transformed back in the format of an image.

The restored image represents the output of filtering the original image by the attribute filter A. For example if the computed attribute is the area of the connected components and if the evaluation criterion is being grater than the threshold $\lambda = 10000$, in the output filtered image, regions whose area are less than 10000 pixels are removed leaving the input image with larger structures. The max-tree representation of an image is suitable for opening and thinning transformations. For closing and thickening, the min-

tree structure is used that can be easily obtained by building the max-tree representation of the input image's complement.

Similar to the definition of morphological profiles in (2.49), attribute profiles can be defined as (2.55)

$$\mathrm{AP}(I) = \mathrm{II}_k : \begin{cases} \mathrm{II}_k = \mathrm{II}_{\gamma}^{T'_\lambda} & \text{with } \lambda = (k-n-1), \quad \forall k \in [n+1, 2n+1] \\ \mathrm{II}_k = \mathrm{II}_{\varphi}^{T'_\lambda} & \text{with } \lambda = (n-k+1), \quad \forall k \in [1, n] \end{cases} \tag{2.55}$$

where II_k is a concatenation of thinning (II_γ) and thickening (II_φ) profiles and T' is a set of ordered criteria. The choice of the attribute filter results in modeling different kinds of features from the image. For example, if we consider the size of the connected components as the attribute, similar to the morphological opening by reconstruction, a multi-scale processing of the input image is performed since we gradually erase bigger structures from the input image. However, if we consider the length of the diagonal of the bounding box of each connected component as the attribute, we are able to compare them with respect to their global extension and this is kind of information that cannot be modeled by morphological filters. Furthermore, if the homogeneity of the pixels in each region is considered as the attribute by computing the pixels' standard deviations, we will be able to perform a multilevel analysis based on the gray level values of the pixels not their size or shapes. In fact, any measure that can be computed for the connected components inside an image may be considered as a morphological attribute. In practice, often two or more of these attributes are combined leading to extract all sorts of useful spatial information from the input image which in this case we call the resulting profile multi-attribute profile (MAP).

2.3.5.5 Extended attribute profiles

Computing the APs explained in Section 2.3.5.4, is applicable only on one band (binary or gray-scale) images. An extension of this algorithm called extended attribute profile was

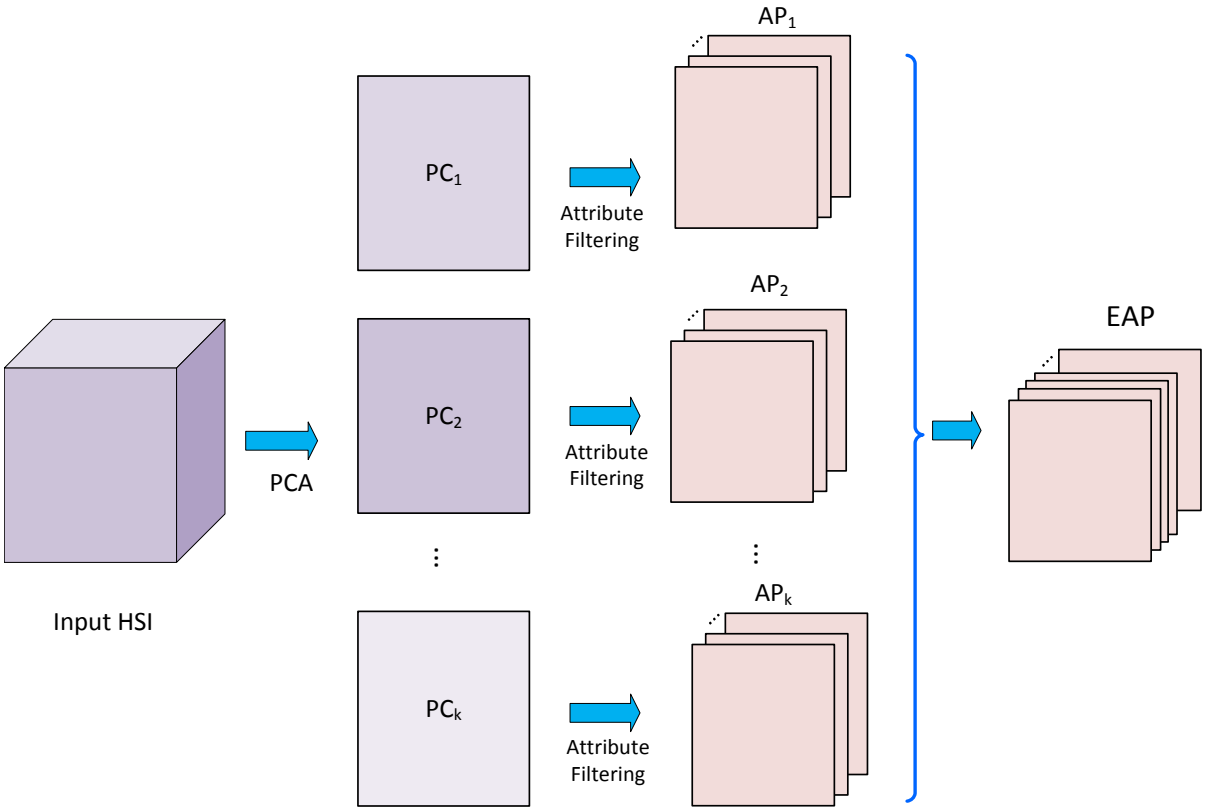


Figure 2.14: Process of obtaining the extended attribute profile from an HSI input where k principal components are preserved in the PCA dimensionality reduction step.

proposed by Dalla Mura *et al.* as a generalized form suitable for multi or hyperspectral images [37]. In their algorithm, similar to the case of EMP, PCA is applied on the input hyperspectral image and the first few PCs are preserved. Then, attribute filters are applied on each PC image separately and the resulting APs are stacked to form the final attribute profile according to (2.56)

$$EAP = \{AP(PC_1), AP(PC_2), ..., AP(PC_k)\} \tag{2.56}$$

where EAP is an extended attribute profile obtained using only one attribute (e.g. area). Figure 2.14 shows a schematic representation of obtaining the EAP from an HSI input.

As it was mentioned earlier at the end of Section 2.3.5.4, usually in practical applications the goal is to extract spatial features from the input scene using multiple attributes

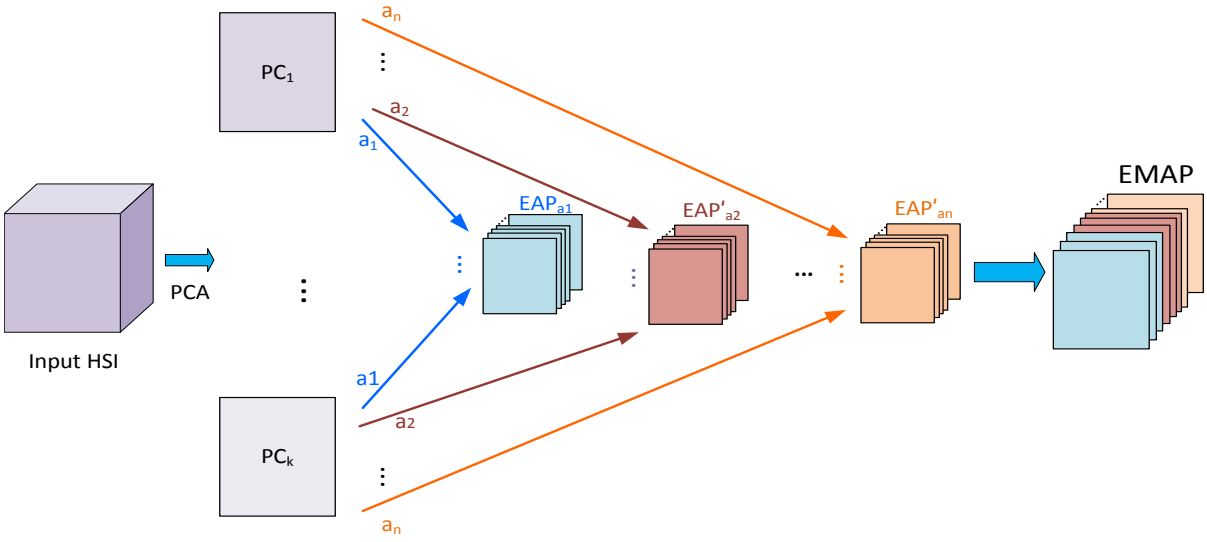


Figure 2.15: Schematic of the step by step process of building the EMAP structure by keeping the first k PCs and using n attribute filters.

filters. In the case of having an HSI as the input image, it can be done by computing extended multi-attribute profiles as is defined in (2.57)

$$EMAP = \left\{ EAP_{a_1}, EAP'_{a_2}, ..., EAP'_{a_n} \right\} \tag{2.57}$$

where a_i represents each of the n attributes used to compute the EMAP data structure. The prime sign indicates the exclusion of the original PC score image in forming the corresponding EAP to avoid the presence of each PC image multiple times in the final EMAP structure. Figure 2.15 shows the process of obtaining the EMAP from an input HSI using k PCs and n attribute filters. Notice that each EAP_{a_i} structure is obtained using the process shown in Figure 2.14. EMAP is a useful method to extract spatial information of remote sensing scenes which usually include structures with different shape, size, and textures. In fact, computing EMAPs of remote sensing scenes gives us important features that helps discriminate between the different classes existing in such scenes.

2.4 Summary

In this chapter, we first presented an introduction on hyperspectral imaging systems. Then, we talked about conventional and the state of the art HSI classification methods. Finally, the last section of this chapter covered some machine learning and image processing methods which we either mentioned in Section 2.2 or will be using in the following chapters.

Chapter 3

Hyperspectral Image Classification Using PCDA and SAE

3.1 Introduction

Deep learning has attracted a lot of attention in the area of image processing recently. Specifically, in the domain of HSI classification, deep learning has shown promising results. One common issue in classification of hyperspectral datasets using deep neural networks is overfitting due to existence of many spectral bands and lack of abundant training samples. To deal with this problem, one common approach is to apply dimensionality reduction methods on hyperspectral datasets [79–82]. DR methods map the input data into a lower dimensional subspace in order to eliminate information redundancy existing in the spectral bands. PCA has been extensively used in the literature as the DR method for HSI classification [41, 42, 45, 54, 57]. For instance in [41] and [42], PCA has been applied to the hypercube before extracting the spatial information of a target pixel. In [45], the input of the CNN is patches extracted from the first few PCs of the original data. Applying PCA on an HSI can be divided into two parts as is depicted in Figures 3.1a and 3.1b. In the first part, the input HSI of size $m \times n \times \lambda$ is converted

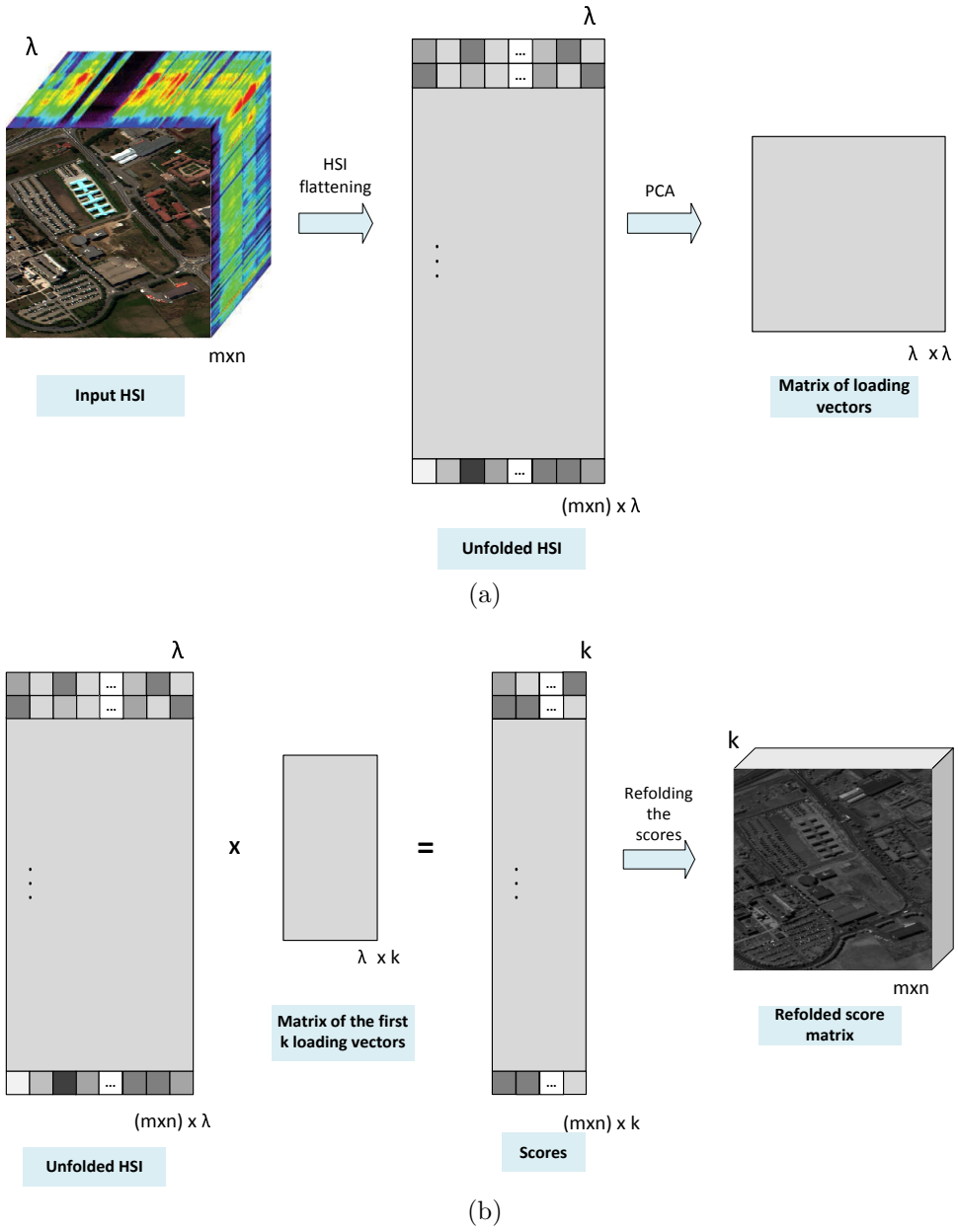


Figure 3.1: Steps of applying PCA on an HSI. (a) Unfolding the input HSI and computing the λ eigenvectors (loading vectors). (b) Multiplying the unfolded HSI by the first k loading vectors and folding the result back in the form of a cube.

(unfolded) to a 2-D matrix of size $(m \times n) \times \lambda$ where m, n, and λ are the number of rows, columns, and spectral bands of the HSI, respectively. Then, PCA is applied on the unfolded HSI which results in a matrix of size $\lambda \times \lambda$ (and also λ eigenvalues which is not shown in the figure) whose columns contain the ordered eigenvectors of its covariance matrix . In the second part, the first k eigenvectors (first k columns of the loading vectors matrix) are retained and the resulting matrix is multiplied by the unfolded HSI. Output of this multiplication will be a matrix of size $(m \times n) \times k$ whose rows are the projections of the spectra of the pixels in the input HSI to the directions of the k first eigenvectors. Finally, in order to have the output in the form of a cube again, this matrix is refolded to a 3-D matrix of size $m \times n \times k$.

Despite its popularity, PCA is an unsupervised DR method. In other words, it does not take into account the class information when mapping original data into a lower dimensional space. Using supervised DR methods such as LDA can possibly improve the classification results. In 2014, Imani *et al.* [83] proposed a new feature extraction method called principal component discriminant analysis (PCDA) to take into account class information while reducing the dimension of the input data. Their results showed that their approach outperformed both PCA and LDA methods. We will describe PCDA method in Section 3.2.2.

In this study, in order to take advantage of both class discriminatory information while performing DR as well as extracting deep spectral-spatial features of the input data, we combined PCDA and deep autoencoder (DAE) [41] methods and we called our method PCDA-SAE. We also performed a thorough search in the hyperparameter space to find the optimal values for the existing hyperparameters. Experimental results on the Indian Pines and University of Pavia datasets demonstrate that our method outperforms both PCDA and DAE methods as well as some of the state of the arts methods even if only a small portion of the data is used for training. Specially, our method improves class-specific accuracies of the classes with very limited training samples.

The rest of this chapter is organized as follows. Section 3.2 presents our proposed

framework for hyperspectral image classification. In Section 3.3 experimental results on the two well-known hyperspectral datasets along with result analysis are presented. Section 3.4 concludes this chapter.

3.2 Method

3.2.1 Proposed framework

The framework of our proposed method, PCDA-SAE, is shown in Figure 3.2. In our method, in order to extract and use deep spectral-spatial features for classification, the information of the neighbors of each target pixel is stacked to its spectrum as suggested in [41]. However, unlike [41] and the majority of the deep learning-based hyperspectral image classification algorithms which use unsupervised dimensionality reduction methods such as PCA, we took advantage of the PCDA method in order to incorporate class information while reducing the dimensionality of the input hyperspectral image. The reason for such a replacement is the fact that PCA maps the input data on the dimensions along which data vary the most without considering class information. In other words, PCA is planned for accurate data representation and not necessarily for data classification. Considering this fact, by obtaining the spatial information by the means of the PCDA method we deliver more representative feature vectors to the SAE network which results in higher classification accuracies as is shown in the next section. Our input feature vector, $\boldsymbol{F}$, to the SAE is then as follows:

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{Y}_{\text{PCDA}} \end{bmatrix} \tag{3.1}$$

where $\boldsymbol{P}$ represents the spectrum vector of each pixel and $\boldsymbol{Y}_{\text{PCDA}}$ is the feature vector obtained from the PCDA method. To obtain $\boldsymbol{Y}_{\text{PCDA}}$, first the dimension of the input HSI is reduced by applying the PCDA method, then a neighborhood area around each target pixel is considered (e.g., a 5x5 square neighborhood as is shown in Figure 3.2).

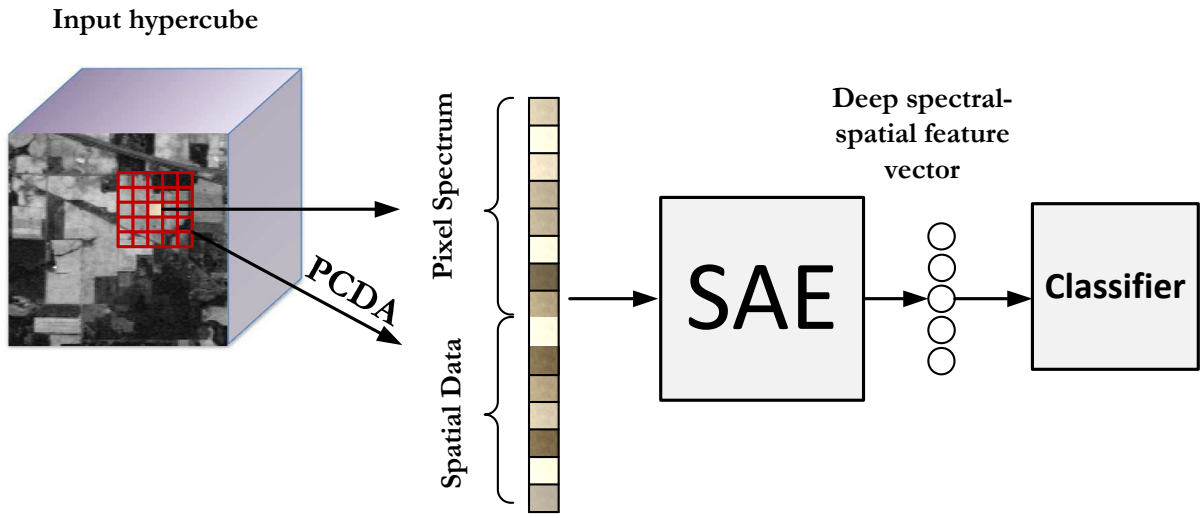


Figure 3.2: Block diagram of the proposed method. PCDA is employed to capture spatial information of each target pixel which then will be stacked with the spectrum of the target pixel to form the input of the SAE network. This network is composed of multiple sparse autoencoders.

Next, the PCDA values of the neighbors are flattened to form the spatial feature vector of the target pixel. To extract deep spectral-spatial features, we employed a deep SAE with sparse autoencoders in its layers. Please refer to Figure 2.8 for a sample AE and SAE. In the following subsection, we explain how the PCDA method works.

3.2.2 Principal component discriminant analysis

PCDA is a combination of PCA and LDA methods [83]. They both are amongst the best well-known DR approaches. However, unlike PCA which is an unsupervised method, LDA considers class information for finding the best subspace for the projection of the input data. The main objective of PCDA is to use PCA components with smaller power as well as the ones with the most variance and utilize possible class discriminatory information of these components by applying LDA on them.

Considering $\boldsymbol{X}$ as the original input of size $d \times N$ where d is the dimension and N is the number of input vectors, PCDA method consists of four major steps: In the first step,

having applied PCA on $\boldsymbol{X}$ and keeping the first n_1 eigenvectors, input data is mapped along these components as follows:

$$\boldsymbol{Y}_1 = \boldsymbol{U}_1 \boldsymbol{X} \quad and \quad \boldsymbol{U}_1 = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{n_1} \end{bmatrix} \tag{3.2}$$

where $\boldsymbol{U}_1$ is an $n_1 \times d$ matrix containing the first n_1 eigenvectors corresponding to the n_1 largest eigenvalues obtained from the input data. Also, $\boldsymbol{Y}_1$ is the input data mapped to the direction of the n_1 principal components and it is of size $n_1 \times N$. Unlike PCA method, in PCDA, components with smaller variances are not discarded. Considering $\boldsymbol{U}_2$ as a $(d - n_1) \times d$ matrix whose rows contain the PCs with smaller variances than the first n_1 eigenvectors, in the second step, original data is mapped along the $\boldsymbol{U}_2$ matrix:

$$\boldsymbol{Y}_2 = \boldsymbol{U}_2 \boldsymbol{X} \quad and \quad \boldsymbol{U}_2 = \begin{bmatrix} \boldsymbol{u}_{n_1+1} \\ \boldsymbol{u}_{n_1+2} \\ \vdots \\ \boldsymbol{u}_d \end{bmatrix} \tag{3.3}$$

where, $\boldsymbol{Y}_2$ is a $(d - n_1) \times N$ matrix containing the data mapped in the direction of the $(d - n_1)$ less powerful eigenvectors. Next, LDA is applied on $\boldsymbol{Y}_2$ in order to capture the useful class information in the data mapped to the direction of the eigenvectors with less variance. To do so, the n_2 first eigenvectors of the $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ matrix is chosen and stored in matrix $\boldsymbol{U}_3$. Where $\boldsymbol{S}_w$ and $\boldsymbol{S}_b$ are the within and between-class scatter matrices of $\boldsymbol{Y}_2$. Finally, $\boldsymbol{Y}_3$ is calculated as follows:

$$\boldsymbol{Y}_3 = \boldsymbol{U}_3 \boldsymbol{Y}_2 \quad and \quad \boldsymbol{U}_3 = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{n_2} \end{bmatrix} \tag{3.4}$$

where $\boldsymbol{U}_3$ is an $n_2 \times (d - n_1)$ matrix whose rows are the first n_2 eigenvectors of the $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ associated with the n_2 largest eigenvalues. Also, $\boldsymbol{Y}_3$ is an $n_2 \times N$ matrix containing $\boldsymbol{Y}_2$ mapped along these n_2 components. The importance of this step is to utilize discrimination information existing in the PCA components with less variance by applying LDA on $\boldsymbol{Y}_2$. PCDA features are built by stacking features obtained in steps one and three as follows

$$\boldsymbol{Y}_{PCDA} = \begin{bmatrix} \boldsymbol{Y}_1 \\ \boldsymbol{Y}_3 \end{bmatrix} \tag{3.5}$$

where $\boldsymbol{Y}_{PCDA}$ is an $(n_1 + n_2) \times N$ matrix and each column of it contains the PCDA features of each input vector. n_1 and n_2 which represent number of retained components in the PCA and LDA steps pf the PCDA method are two hyperparameters of the model that need to be determined. We will show in Section 3.3.2 how we choose these values. In the next section, we will see how combining features obtained by (3.5) with the SAE network improves classification accuracies.

3.3 Experimental results

In this section, we evaluate the performance of our method and compare it to several hyperspectral image classification algorithms. The metrics used for the performance evaluation are class-specific accuracy, overall accuracy (OA), average accuracy (AA), and Kappa coefficient. We have used two well-known online HSI datasets including Indian Pines and University of Pavia scenes[1]. All experiments are performed using Matlab R2017a on a desktop with an Intel Core i5 2.7GHz cpu and 8GB RAM.

3.3.1 Data description

1) Indian Pines: This dataset has been collected by an airborne visible/infrared imaging spectrometer (AVIRIS) sensor over the Indian Pines site in North-western Indiana, USA.

[1] `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`

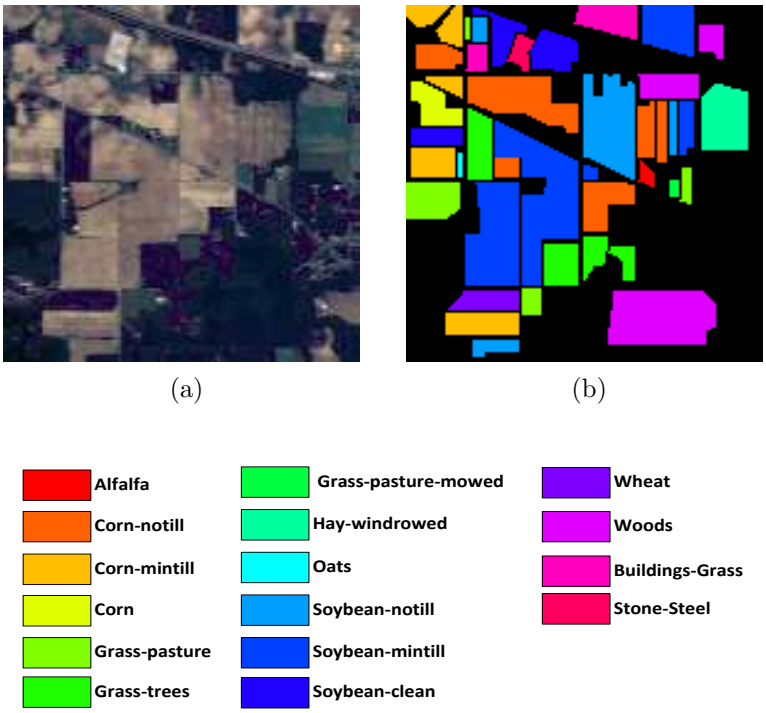


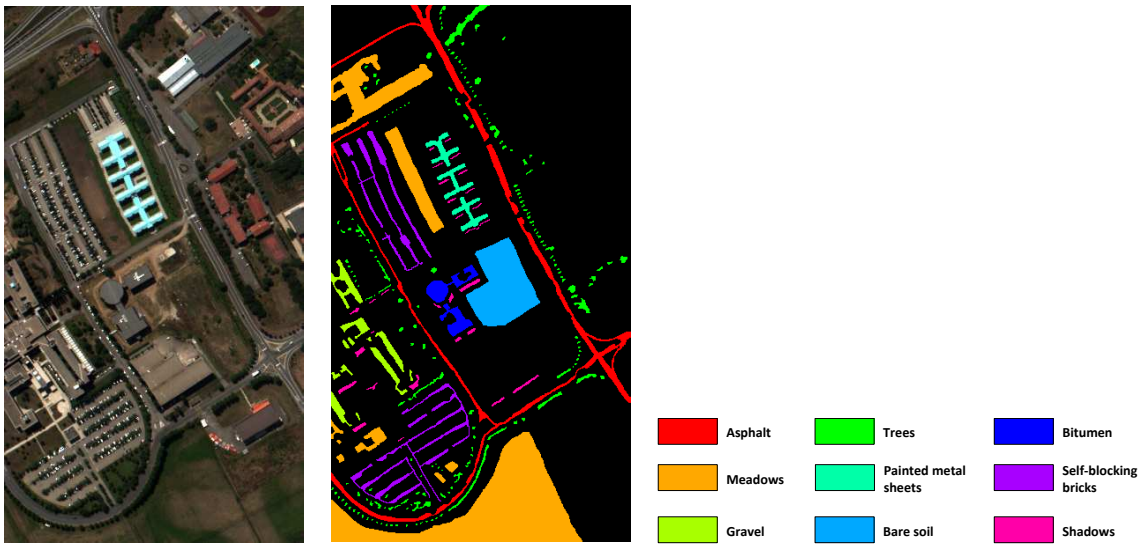
Figure 3.3: Indian Pines dataset. (a) False color image and (b) pseudo ground truth image.

The wavelength range of this hyperspectral database is 0.4-2.5μm. There are 224 spectral bands in this HSI and each band contains a 145×145 image. Having removed water absorption bands, number of spectral bands was reduced to 200. A false color image and the ground truth map of the Indian Pines dataset containing 16 different classes is shown in Figure 3.3. Also, Table 3.1 lists the number of labeled samples for each class in this dataset.

2) University of Pavia: This database has been collected by a reflective optics system imaging spectrometer (ROSIS) sensor over Pavia, northern Italy. University of Pavia hyperspectral dataset has 103 spectral bands in the wavelength range of 0.43-0.86 μm with each band having the spatial extent of 610×340 pixels. Ground truth of this database contains 9 different classes. True color image, ground truth map, and number of labeled samples of this HSI are shown in Figures 3.4a, 3.4b, and Table 3.2, respectively.

Table 3.1: Number of labeled samples for the different sixteen classes of the Indian Pines dataset

No	Class	Available data
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93



(a) (b)

Figure 3.4: University of Pavia dataset. (a) True color image and (b) pseudo ground truth image.

Table 3.2: Number of labeled samples for the different nine classes of the university of Pavia dataset

No	Class	Available data
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Tress	3064
5	Painted metal sheets	1345
6	Bare Soil	5029
7	Bitumen	1330
8	Self-Blocking Bricks	3682
9	Shadows	947

3.3.2 Parameter tuning

Although the network's weights are learned during the training, there are hyperparameters in our method that need to be carefully tuned. These hyperparameters include: spatial extent of the neighborhood area of the target pixel when extracting the spatial information (neighborhood size), number of retained principal components in PCA and LDA steps of the PCDA method, (n_1 and n_2, respectively), and number of neurons in each autoencoder (number of hidden units). We used a SAE with two hidden layers in its architecture. The reason behind employing only two hidden layers is to keep the number of trainable parameters low and prevent our deep learning model to overfit to the available training data.

To see how changing the mentioned hyperparameters affects the OA, we tried different values and trained the network for each combination on the training set and evaluated the trained network on the test set. Since a deep neural network requires many training samples to be trained effectively, in this set of experiments, in order to let the SAE be trained with sufficient training data, we used 50% of the data for training and 50% for the test. So, we split the labeled samples randomly into two sets of training and test with the ratio of 1:1. All sets of experiments are repeated ten times using randomly selected labeled samples and the average value of each accuracy metric alongside with

the corresponding standard deviation is reported. Considering the spatial resolution of the datasets used in this chapter, we used three different values for the neighborhood region size including 3×3, 5×5, and 7×7 pixels. These values are consistent with the neighborhood sizes normally used in the literature to extract the spatial information of a target pixel. Although one can use regions greater than 7×7 pixels, it may not be very suitable particularly for classes that own small patches of pixels in the image. Considering a large square neighborhood area around a target pixel in such patches results in incorporating spectral information of the neighbors that do not necessarily share the same class label as the target pixel. Although using larger values for the neighborhood size can still be beneficial for the classes that are represented with large and wide patches in the image, limiting this quantity to the value of 7, increases the generalization power of our model.

Since in PCA and LDA methods the first few components preserve the most variations in the dataset, we used 5 different values (1 to 5) for the n_1 and n_2 hyperparameters. We trained the SAE for 75 ($3 \times 5 \times 5$) different combinations of these three hyperparameters. For this set of experiments, we used a two-layer SAE with 60 hidden units in the first and 30 units in the second sparse AE.

Distribution of the OA obtained by our method on Indian Pines and University of Pavia datasets using different values for the three hyperparameters are shown in Figures 3.5 and 3.6, respectively. Also, Figures 3.7 (a) and (b) demonstrate the distribution of the OA using different values for the n_1 and n_2 and the neighborhood size of 7×7. As can be seen from Figures 3.5 and 3.6, a larger neighborhood size results in higher classification accuracies for both datasets. Also in general, higher values for the n_1 and n_2 increases the OA. For example, for the two datasets, keeping 5 components in the LDA step of the PCDA method (i.e., $n_2 = 5$) results in higher classification accuracies and this is because of the fact that retaining more components enhances the spatial feature vector. Another common observation is the increase in OA with the increase in the number of components in the PCDA method. We also observed that for the University of Pavia dataset, for a fixed number of retained components, keeping more components

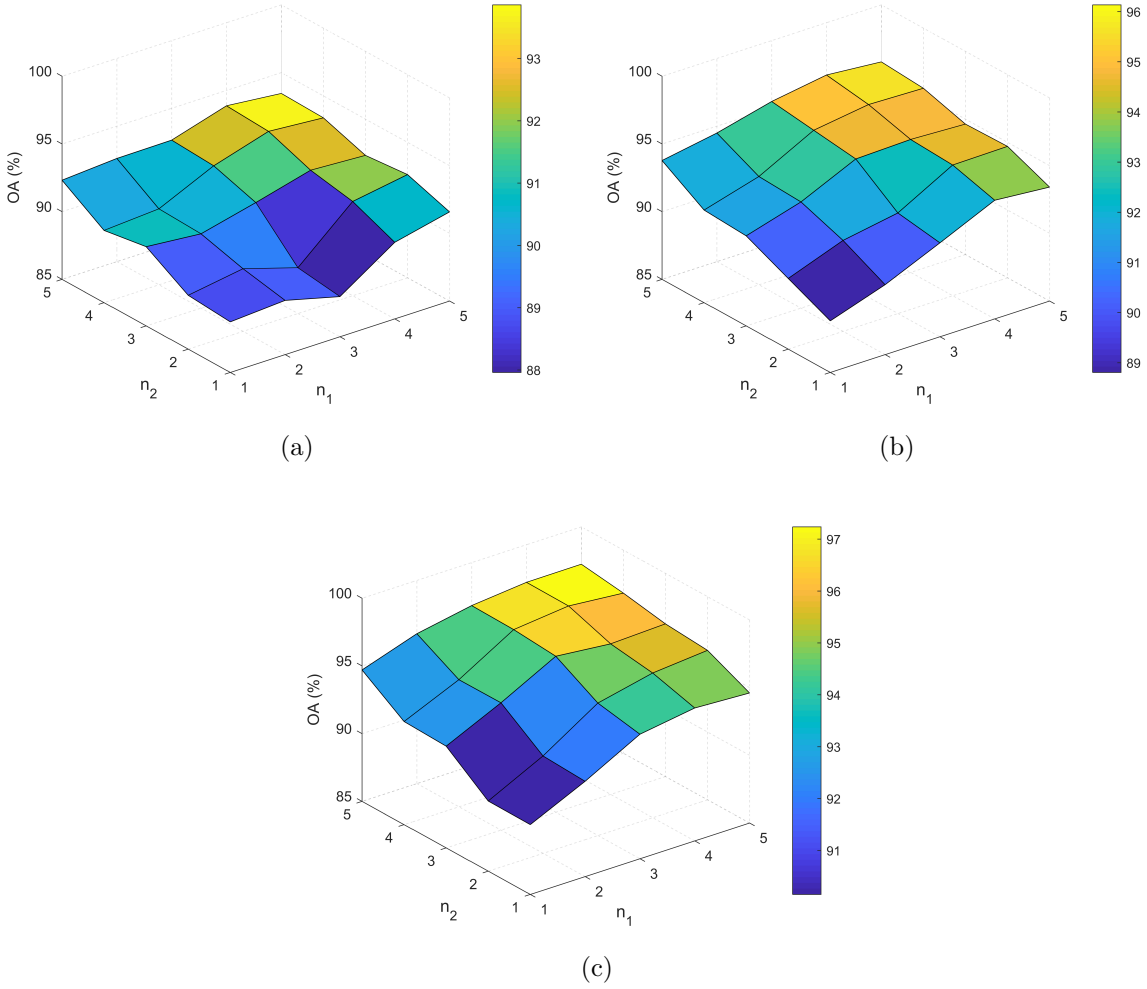


Figure 3.5: Distribution of the OA obtained by our method with different values for the n_1 and n_2 for Indian Pines dataset using neighborhood sizes of (a) 3×3, (b) 5×5, and (c) 7×7.

in the LDA step, than PCA step results in better OAs . On the one hand, keeping more components results in higher accuracies. On the other hand, it increases the size of the input of the deep network and consequently rises computational cost. Therefore, to have a trade off between the accuracy and computational cost we have chosen $n_1 = 3$ and $n_2 = 4$ for the Indian Pines and $n_1 = 2$ and $n_2 = 3$ for the University of Pavia datasets, respectively. Also, for both datasets neighborhood size is chosen to be 7×7 pixels. For the number of hidden units for each AE, we tried 50-120 units and 40-110 units with the steps of 10 for the Indian Pines and University of Pavia datasets, respectively. Figures

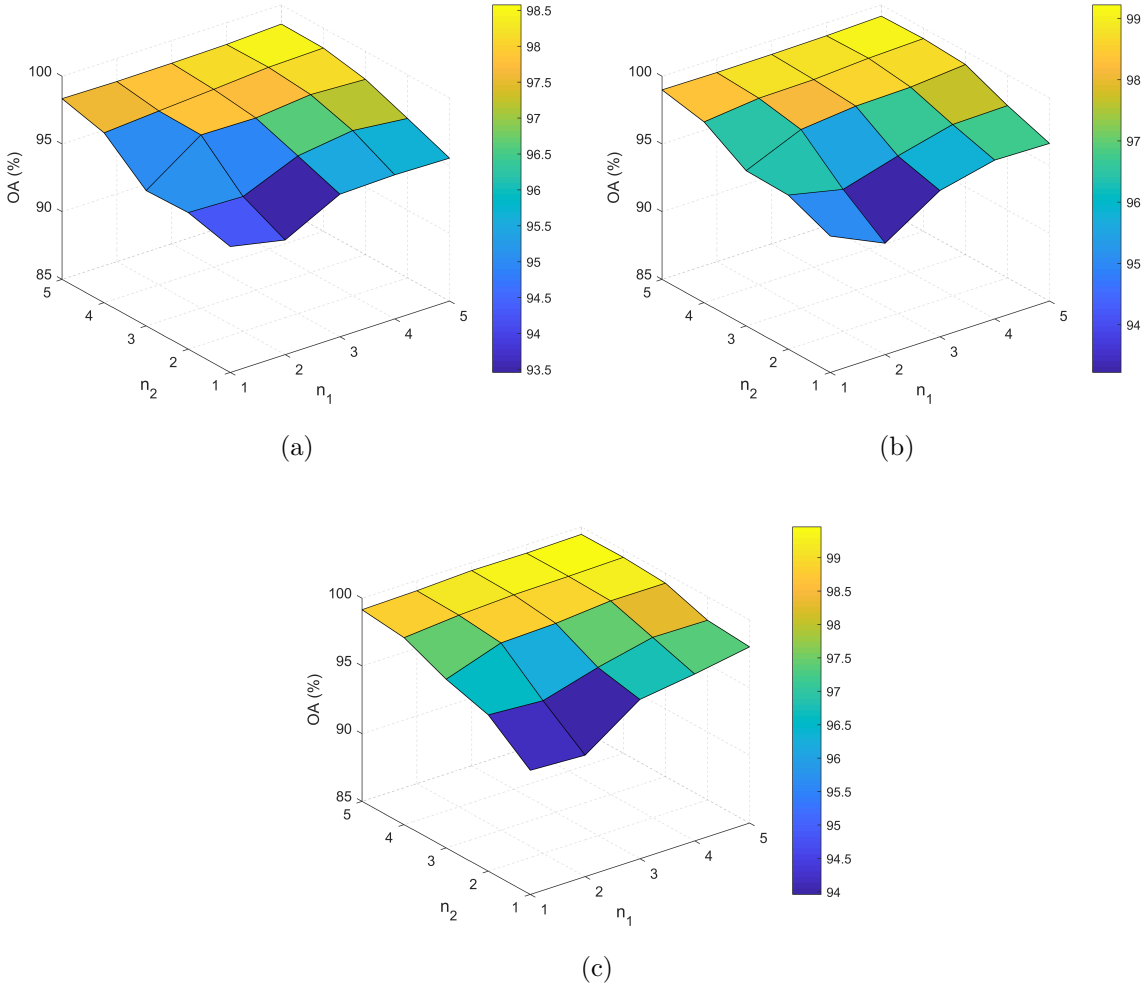


Figure 3.6: Distribution of the OA obtained by our method with different values for the n_1 and n_2 for University of Pavia dataset using neighborhood sizes of (a) 3×3, (b) 5×5, and (c) 7×7.

3.8 (a) and (b) show the distribution of the OA for the different number of hidden units for the two datasets. As can be seen from these figures, as the number of hidden units increases, the OA also generally increases. However, since the performance seems to be very close for the different values of this hyperparameter, we chose hidden unit number of 80 for both datasets.

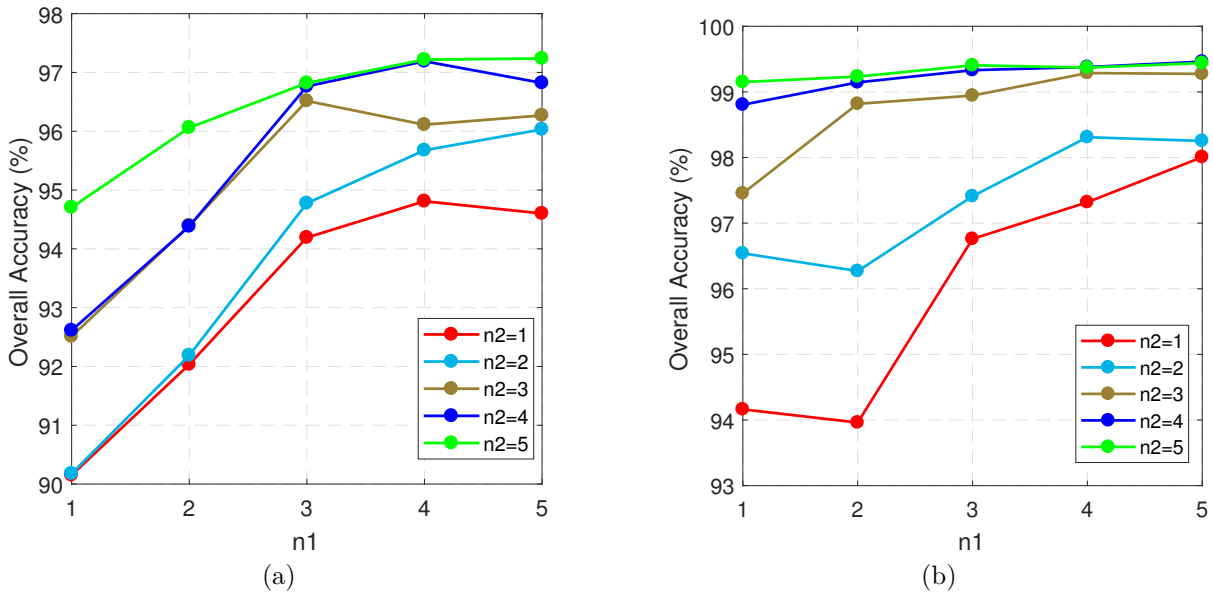


Figure 3.7: Distribution of the OA obtained by our method using different values for the n_1 and n_2 for (a) Indian Pines and (b) University of Pavia datasets.

3.3.3 Performance evaluation

To check the effectiveness of our proposed approach, we compared it with several classification methods such as support vector machine (SVM), principal component discriminant analysis (PCDA) [83], deep auto-encoder (DAE) with joint features [41], EMAP, CNN-PPF [47], and EMAP-SAE [48]. We implemented both linear and kernel SVM using *libsvm* library [84] and used gaussian radial basis function (RBF) kernel for the non-linear SVM. In the experiments, the input of the SVM classifier is the spectrum of each pixel. Also, we have used RBF SVM with the PCDA and EMAP feature extraction methods. To find the best values for the parameters of the SVM classifier wherever it is employed in this chapter, we performed 10-fold cross validation and the optimal values are listed in Table 3.3 for both datasets. For the DAE, EMAP-SAE, CNN-PPF, and our method LR classifier with the softmax activation is used.

PCDA is implemented based on [83] and n_1 and n_2 are set to 8. For the DAE, similar to PCDA-SAE, we employed 3×3, 5×5, and 7×7 for the neighborhood size. However, since the only dimensionality reduction method used in DAE is PCA, 10 values (1 to

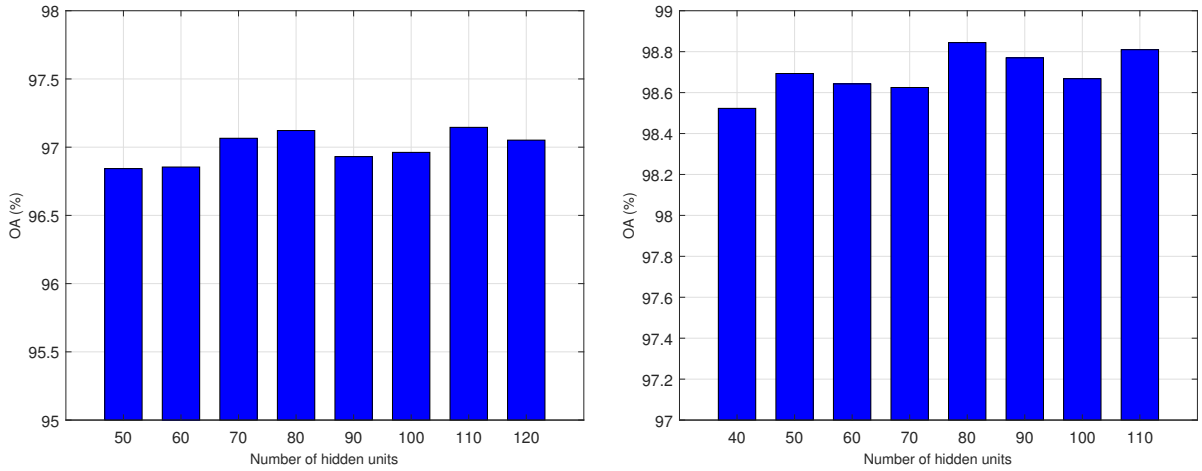


Figure 3.8: Distribution of the OA of our method vs different values of the number of hidden units for (a) Indian Pines and (b) University of Pavia datasets.

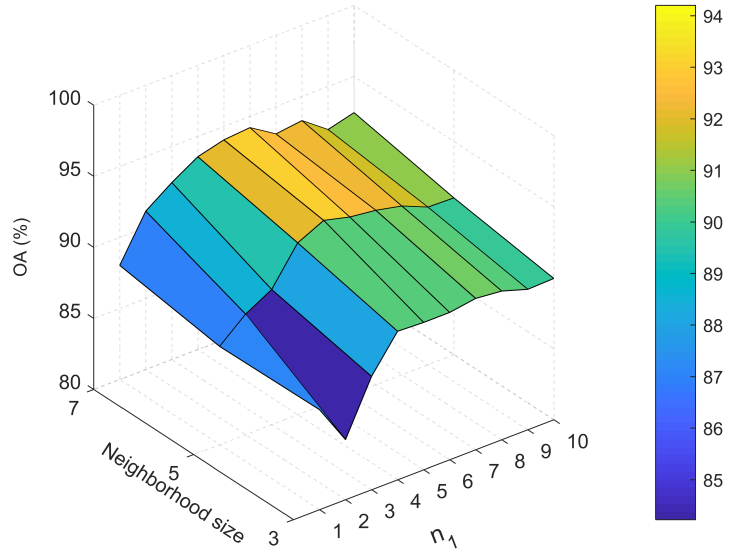


Figure 3.9: Distribution of the OA obtained by DAE-LR using different values for n_1 and the neighborhood size for the Indian Pines dataset.

10) is used for the n_1 hyperparameter which in this case represents number of retained components in the PCA step. Figures 3.9 and 3.10 show the OA obtained by DAE-LR method on the Indian Pines and University of Pavia datasets for different values of neighborhood size and n_1. As can be seen from Figure 3.9, increasing n_1 form 1 to 6 results in improving the OA. However, increasing this parameter after 6 has an opposite

Table 3.3: Best values for the parameters of the SVM classifiers used in this study after performing 10-fold cross validation.

	Databases			
Methods	Indian pines		University of Pavia	
Linear SVM	$c = 10^2$		$c = 10^1$	
RBF SVM	$c = 10^2$,	g =1	$c = 10^3$,	$g = 0.1$
PCDA-RBF SVM	$c = 10^5$,	g =0.1	$c = 10^6$,	$g = 0.01$
EMAP RBF SVM	$c = 10^2$,	g =0.1	$c = 10^4$,	$g = 0.001$

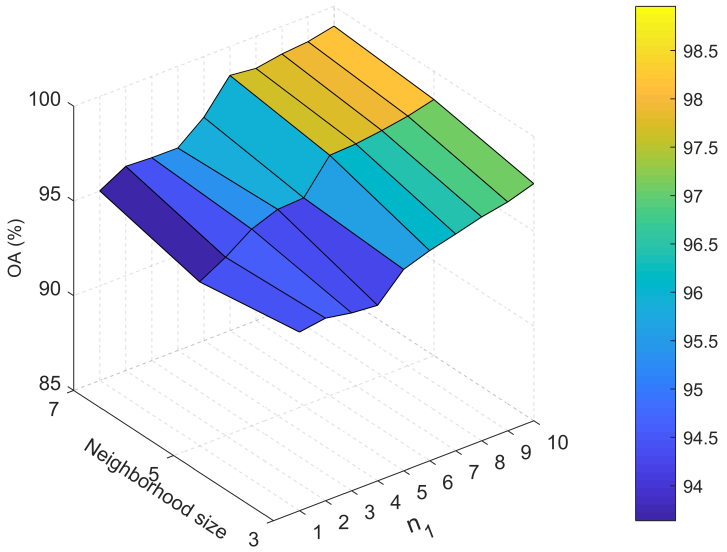


Figure 3.10: Distribution of the OA obtained by DAE-LR using different values for n_1 and the neighborhood size for the University of Pavia dataset.

effect. So, n_1=6 is chosen to be the optimal value for the Indian Pines dataset. For the University of Pavia dataset, this parameter is chosen to be 5 to be consistent with the total number of components used in PCDA-SAE. Also, for both datasets, neighborhood size of 7×7 is used.

Class-specific accuracies, OA, AA, and Kappa coefficient obtained by all the methods experimented in this chapter and for the two hyperspectral datasets are listed in Tables 3.4 and 3.5. In this set of experiments 50% of the samples are used for training. As can be seen from these tables, PCDA-SAE outperforms all other methods in terms of the all accuracy metrics. For example according to Table 3.4, the OA obtained by our method is

Table 3.4: Classification accuracies (%) of different methods for Indian Pines dataset using 50% of the training data.

Class	Linear SVM	RBF SVM	PCDA-SVM	EMAP-SVM	DAE-LR	EMAP-SAE	PCDA-SAE
1	79.32	83.48	83.48	90.87	87.46	82.61	97.75
2	82.69	86.61	85.71	84.37	93.14	94.06	95.76
3	70.21	81.71	71.83	94.51	92.74	92.82	96.50
4	64.21	78.99	74.79	96.72	85.20	92.35	95.58
5	94.96	95.87	93.10	94.01	96.33	94.92	98.57
6	97.97	97.62	96.74	98.25	99.29	99.09	99.60
7	75.14	86.43	84.29	87.86	86.94	80.71	96.27
8	98.92	98.70	98.12	100	99.41	99.54	99.91
9	75.68	75.00	79.00	90.00	94.61	80.00	96.18
10	72.59	87.00	80.60	78.85	93.72	92.28	95.67
11	79.38	90.59	86.79	97.63	94.96	95.88	96.04
12	78.16	87.27	82.79	94.07	90.26	91.72	96.87
13	98.54	98.83	98.35	97.96	98.04	98.93	99.79
14	94.35	96.70	96.90	99.62	98.30	98.44	99.55
15	71.48	71.81	66.63	99.74	93.24	94.66	96.24
16	93.25	95.11	91.28	97.87	98.79	90.64	98.21
OA	82.99±0.60	89.82±0.40	86.59±0.49	93.77±0.18	94.87±0.70	95.22±0.32	**97.12±0.43**
AA	82.93±1.83	88.23±0.98	85.65±1.44	93.90±1.26	93.90±1.06	92.42±1.38	**97.41±0.66**
Kappa	0.81±0.01	0.88±0.005	0.85±0.006	0.93±0.002	0.94±0.008	0.94±0.004	**0.97±0.005**

Table 3.5: Classification accuracies (%) of different methods for University of Pavia dataset using 50% of the training data

Class	Linear SVM	RBF SVM	PCDA-SVM	EMAP-SVM	DAE-LR	EMAP-SAE	PCDA-SAE
1	90.42	91.88	94.12	98.34	96.78	97.29	98.76
2	95.97	97.28	97.71	99.64	99.32	98.80	99.84
3	71.11	75.96	72.57	98.91	87.22	93.01	93.80
4	91.62	93.90	93.04	90.37	98.30	97.75	99.17
5	99.85	99.88	99.85	99.45	99.85	99.67	99.91
6	56.67	79.06	90.02	98.78	95.67	95.52	99.42
7	78.00	82.90	85.20	97.61	84.23	93.65	96.00
8	81.65	83.51	90.17	95.78	94.43	94.46	96.10
9	94.10	96.27	99.79	71.6	99.40	99.86	99.77
OA	87.26±0.19	91.44±0.10	93.76±0.10	97.61±0.41	96.96±0.58	97.34±0.11	**98.84±0.15**
AA	84.38±0.27	88.96±0.29	91.39±0.23	94.49±0.32	95.02±0.96	96.67±0.21	**98.09±0.25**
Kappa	0.83±0.002	0.88±0.001	0.92±0.001	0.96±0.002	0.96±0.008	0.96±0.002	**0.98±0.002**

by 2.25% and 1.90% higher than those obtained by DAE-LR and EMAP-SAE methods, respectively. Also, for the University of Pavia dataset the OA obtained by our method is superior to DAE-LR and EMAP-SAE by 1.88% and 1.50%, respectively. Moreover, according to these tables our method results in lower variance in class-specific accuracies for both datasets. In other words, our method delivers high classification accuracies for all classes in the dataset. For example, based on the last column of Table 3.4, the minimum and maximum class-specific accuracies obtained by our method and EMAP-SAE are [95.58%, 99.91%] and [80.00%, 99.54%], respectively. The difference of these limits for our method is 4.33% which is remarkably lower than the 19.54% difference obtained by EMAP-SAE method. Similar argument can be made for the University of Pavia dataset as well.

Insufficiency of the labeled data to train a classifier is a common issue in the domain of remote sensing hyperspectral image classification. Therefore, it is important to check the robustness of a classifier with respect to the size of the training data. Therefore, we also trained our model with a different ratio of train and test samples. For the Indian Pines dataset, we randomly chose 20% of each class for training and the remaining 80% for the test. However, this dataset contains classes with very few labeled samples such as Alfalfa, Grass-pasture-mowed, Oats, and Stone-Steel-Towers compared to other classes. Therefore, in order to reduce the effect of having an unbalanced dataset in the results, in this case, we used half of the data for training and the rest half for the test.

For the University of Pavia dataset 10% of the labeled samples of each class is used for training and the rest 90% for the test. Details of the number of train and test samples for each dataset are given in Tables 3.6 and 3.7. To make fair comparisons, the same amount of data is used in order to train the other models as well. As can be seen from these tables, PCDA-SAE outperforms other methods in terms of OA, AA, and Kappa coefficient, and class-specific accuracies for the majority of classes. The last row of Tables 3.6 and 3.7 lists the required test times for the different methods. As can be seen, the test time of our method is relatively lower than the other methods which makes it a suitable choice for possible real time applications.

Table 3.6: Classification accuracies (%) and the test time (s) of the different methods on Indian Pines dataset using 20% of the labeled samples for training.

Class	No. of samples		Methods						
	Train	Test	SVM	PCDA-SVM	DAE-LR	EMAP-SVM	CNN-PPF LR	EMAP-SAE	PCDA-SAE
1	23	23	84.35	84.78	91.74	92.17	90.43	80.87	98.26
2	286	1142	81.50	82.86	83.91	79.84	91.30	88.04	87.59
3	166	664	75.83	68.64	82.86	91.39	83.73	87.86	90.69
4	47	190	65.42	61.89	76.58	92.05	84.05	87.37	84.63
5	97	386	92.62	90.41	94.35	84.66	93.70	92.77	96.27
6	146	584	96.37	94.59	98.34	96.46	99.67	97.58	99.49
7	14	14	88.57	87.86	89.29	90.71	84.28	87.86	95.00
8	96	382	96.52	95.31	98.69	99.87	99.45	96.23	99.79
9	10	10	79.00	74.00	95.00	97.00	68.00	87.00	100
10	194	778	81.23	75.12	87.04	74.88	87.18	87.39	89.37
11	491	1964	86.25	85.01	89.45	96.69	93.66	93.95	91.22
12	119	474	80.78	74.83	74.18	91.81	90.67	84.77	92.87
13	41	164	97.80	94.09	98.41	97.87	98.29	96.28	99.57
14	253	1012	96.13	96.17	96.15	98.85	97.74	97.67	98.22
15	77	309	60.45	54.50	85.60	97.96	71.59	91.84	91.84
16	46	47	94.04	94.04	99.36	100	99.57	90.63	98.51
OA	2106	8143	85.47 ±0.34	83.18 ±0.53	88.96 ±0.50	91.32 ±0.60	91.95 ±0.98	91.97 ±0.53	**92.81 ±0.12**
AA			84.80 ±1.08	82.13 ±1.26	90.06 ±0.96	92.64 ±0.93	89.58 ±0.93	90.51 ±1.11	**94.58 ±0.16**
Kappa			0.83 ±0.004	0.81 ±0.006	0.87 ±0.006	0.90 ±0.007	0.91 ±0.011	0.91 ±0.006	**0.92 ±0.001**
Test time			3.20	0.49	0.12	1.14	2.41	0.86	0.11

Classification maps obtained by the methods investigated in this chapter are illustrated in Figures 3.11 and 3.12 using 20% and 10% training data for the Indian Pines and University of Pavia datasets, respectively. As can be seen from these figures consistent with the results presented in Tables 3.6 and 3.7, our method produces smoother output maps which for the majority of classes include fewer misclassified pixels.

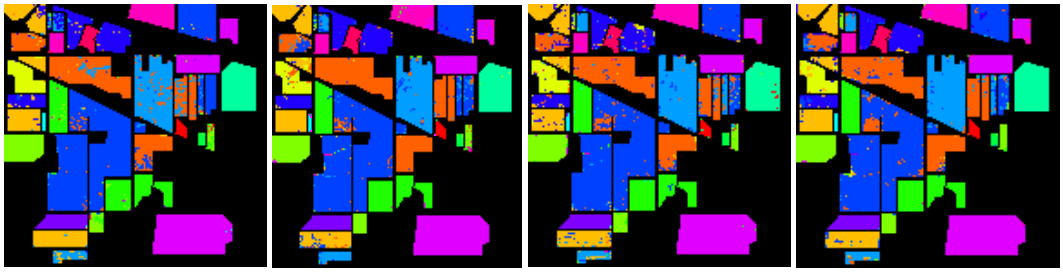
Table 3.7: Classification accuracies (%) and the test time (s) of the different methods on University of Pavia dataset using 10% of the labeled samples for training.

Class	No. of samples		Methods						
	Train	Test	SVM	PCDA-SVM	DAE-LR	EMAP-SVM	CNN-PPF LR	EMAP-SAE	PCDA-SAE
1	663	5968	89.17	93.09	94.07	90.08	98.82	95.32	96.63
2	1865	16784	96.32	97.25	98.70	97.85	99.28	97.64	99.55
3	210	1889	71.97	70.96	84.43	93.75	82.30	92.79	85.61
4	306	2758	92.32	91.07	95.92	96.36	93.22	94.76	97.68
5	134	1211	99.47	99.71	99.45	99.31	99.73	99.30	99.20
6	503	4526	72.67	86.00	89.69	91.07	95.42	91.38	98.42
7	133	1197	75.23	82.68	82.53	89.66	87.05	89.49	89.90
8	368	3314	75.98	88.89	90.06	94.51	92.58	91.40	90.62
9	95	852	94.84	99.68	98.23	77.89	99.42	99.29	99.55
OA/Total	4277	38499	88.61 ±0.17	92.51 ±0.12	94.79 ±0.43	94.60 ±0.07	96.55 ±0.32	95.40 ±0.30	**97.07 ±0.16**
AA			85.33 ±0.45	89.93 ±0.16	92.56 ±0.64	92.28 ±0.26	94.20 ±1.10	94.60 ±0.46	**95.24 ±0.24**
Kappa			0.85 ±0.002	0.90 ±0.002	0.93 ±0.006	0.93 ±0.001	0.95 ±0.004	0.94 ±0.004	**0.96 ±0.002**
Test time			10.09	1.80	0.36	5.09	8.02	0.46	0.34

3.4 Conclusion

In this chapter, a method for classification of hyperspectral images is proposed. In order to exploit useful class information while performing dimensionality reduction as well as deep spectral-spatial features of the input data, PCDA and DAE methods are combined. In our model, SAE is composed of layers of sparse autoencoders. The resulting method, called PCDA-SAE, is applied on the Indian Pines and University of Pavia datasets. In the experiments, we used two different ratios of training and test samples and four accuracy metrics OA, AA, kappa coefficient, and class-specific accuracy. In order to find the best values for the hyperparameters of our model we did a thorough search in the model's hyperparameter space and through extensive experiments trained over one hundred networks for each of the two datasets used in this study. The experimental results demonstrated that PCDA-SAE improves the classification results of the PCDA

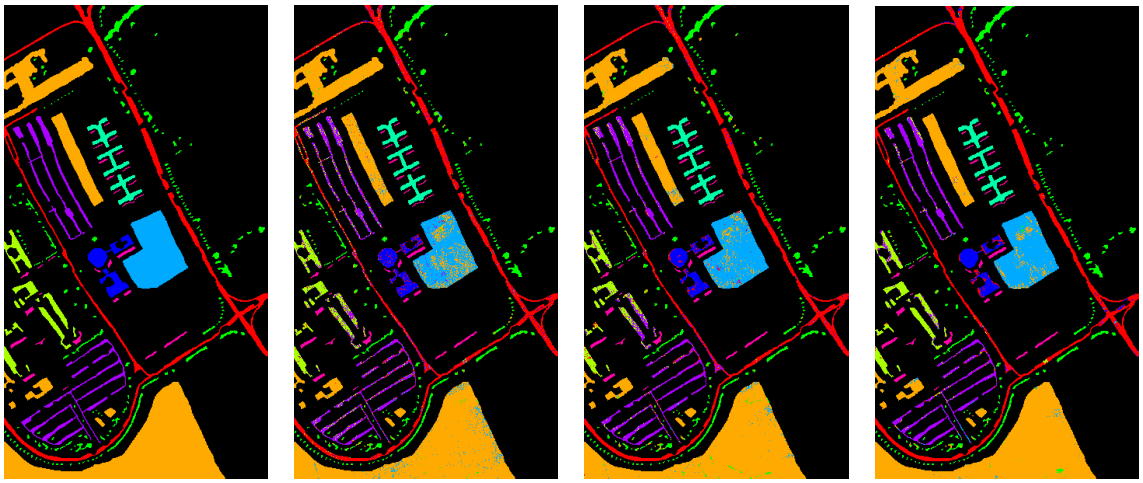
(a) (b) (c) (d)



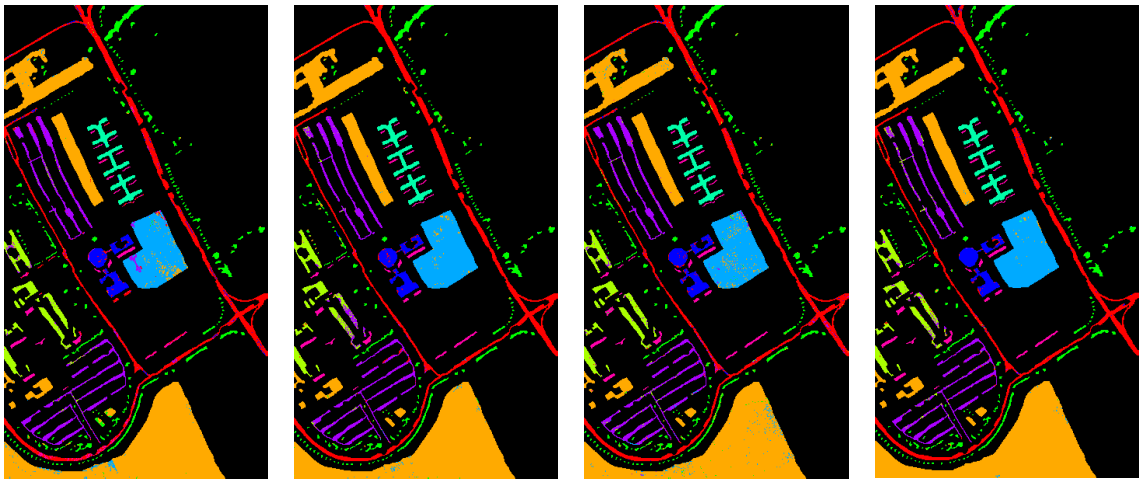
(e) (f) (g) (h)

Figure 3.11: Indian Pines.(a) Ground truth and classification maps obtained from different methods using 20% of labeled data for training. (b)RBF-SVM, (c) PCDA-SVM, (d) DAE-LR, (e) EMAP-SVM, (f) CNN-PPF-LR, (g) EMAP-SAE, and (h) PCDA-SAE.

and DAE methods as well as some conventional and recent deep-learning based HSI classification algorithms in terms of all the accuracy metrics used even if only 10% or 20% of the labeled data is used for training. Moreover, the test time associated with our proposed method is less than those of the other methods which makes our method more suitable for real-time remote sensing applications.



(a) (b) (c) (d)



(e) (f) (g) (h)

Figure 3.12: University of Pavia.(a) Ground truth and classification maps obtained from different methods using 10% of labeled data for training. (b)RBF-SVM, (c) PCDA-SVM, (d) DAE-LR, (e) EMAP-SVM, (f) CNN-PPF-LR, (g) EMAP-SAE, and (h) PCDA-SAE.

Chapter 4

Distance transform based spectral-spatial feature vector for HSI classification with SAE

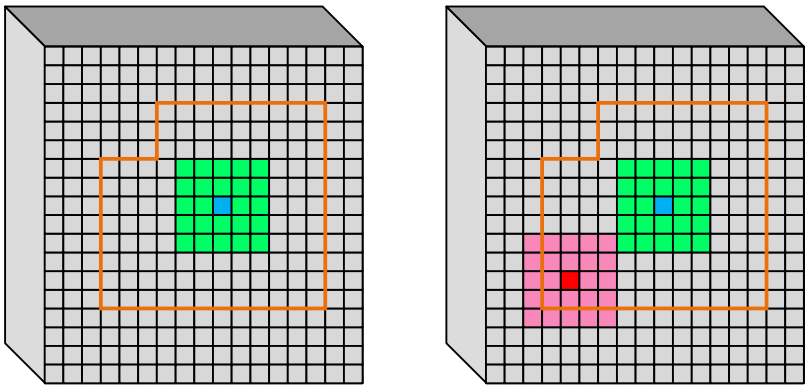
4.1 Introduction

Pixel-wise classification of hyperspectral images is a common practice in the remote sensing domain. Recently, deep learning based approaches have attracted a lot of attention among researchers. One of the significant factors to enhance the classification accuracy of remote sensing hyperspectral image datasets is to effectively combine spectral and spatial information to form the input feature vector to the deep network. In fact, in addition to the spectral data, spatial location of the pixels provide a valuable source of information which improves the classification results. Exploiting the spatial contextual information from an HSI is generally done by two types of methods: In the first type, spatial information are stacked to the spectrum of each pixel and a unified joint feature vector is fed to the network [41, 42, 48]. In the other kind, however, spatial information are used during the test phase to reduce the noise and smooth the primary classification

map [47].

In both types of approaches, pixels are not considered as individual samples, but their location in the image and the surrounding neighbors help identify their labels. But should all the neighbors contribute the same when collecting the spatial information of the target pixels? In this chapter, we propose a joint spectral-spatial feature extraction method that assigns different weights to the neighboring pixels depending on how far they are from the edges in the image. In other words, the location of pixels with respect to the edges in the HSI determines how much they contribute in forming the spatial feature vector. To obtain such a distance measure, having calculated the gradient image of the input HSI, we measured the distance transform of all pixels with respect to the dominant edge pixels in the image. These distance transform values are obtained by the means of our proposed method for finding a distance transform image from an input HSI. These values are then serve as extra features to each pixel's spatial feature vector. Furthermore, in order to add geometric attributes, we also integrated EMAP features [37] to the spatial feature vector. The joint spectral-spatial feature vector is given to a two layer stacked autoencoder including a sparse AE in each layer. Finally, in order to see the effect of using class discriminatory information, we reduced the spectral dimension of the input HSI datasets with the PCDA [83] method as discussed in Chapter 3 as well as PCA approach and applied our proposed distance transform-based spatial feature vector on the resulting hyprecube. We performed extensive experiments on three hyperspectral datasets (Salinas, University of Pavia, and Surrey) and results show that our proposed method achieves higher classification accuracies compared to some traditional as well as recent deep learning based algorithms.

This chapter is organized as follows: Section 4.2 presents our proposed feature extraction and classification framework. Experimental results are presented in Section 4.3 and Section 4.4 concludes this chapter.



(a) (b)

Figure 4.1: Schematic of the justification of using the distance transform in the spatial feature vector.

4.2 Methodology

The idea behind using spatial information as well as spectral information in classification of hyperspectral images is the fact that in an HSI, adjacent pixels belong to the same class with a high probability [43]. This makes sense if these neighboring pixels are far from the edges in the image. As we get closer to an edge the probability that adjacent pixels belong to different classes increases. Consider cases shown in Figure 4.1. In Figure 4.1a, the orange border encompasses group of pixels that all belong to the same class. The blue square is a pixel located not close nor on the border of the area outlined with the orange line. Green squares depict the neighborhood area around the blue pixel. Due to the spatial location of the blue pixel, it sounds reasonable to assume that this pixel and its neighbors belong to the same class, so when forming the feature vector of the blue pixel, we consider the spectral values of the green region as features of the blue pixel (since they all have a same label).

Lets consider another case when the target pixel is close or on the border. Such a pixel along with its neighborhood are depicted as red and pink in Figure 4.1b, respectively. In this case, not all neighbors belong to the same class as the red pixel, so, the distance

of pixels to the border seems to be important. We would like to decrease the effect of the neighbors in the target pixel's spatial feature if they are close or on an edge. Therefore, we present the value of each neighbor along with its distance transform value. The further the neighbor is located with respect to an edge the larger its corresponding distance transform value becomes and vice versa. In order to find the distance transform image from an HSI cube, we propose the following procedure: First, to remove noise, we applied Gaussian smoothing filter on each band of the input HSI. Then, we calculated the gradient image by employing the method described in [85] as follows:

- First, for each spectral band, we computed the image gradients at the four major directions using Sobel filters [86] according to (4.1). These directions are the ones corresponding to horizontal edges (0°), vertical edges (90°), and the two diagonal edges (45°and 135°).

$$\begin{aligned} \boldsymbol{G}_{ij} &= \boldsymbol{I}_j * \boldsymbol{h}_i \\ s.t \quad i &\in \{0, 45, 90, 135\} \quad and \quad j = 1, 2, ..., N \end{aligned} \tag{4.1}$$

  where $\boldsymbol{I}_j$ is the image at band j, $\boldsymbol{h}_i$ is the Sobel filter at one of the four specified directions, and $\boldsymbol{G}_{ij}$ is the gradient image of band j obtained from the ith Sobel filter. Also, $*$ represents the convolution operation.

- Next, in order to obtain the single image gradient for each direction, the corresponding gradients of all bands are added together

$$\boldsymbol{G}_i = \sum_{j=1}^{N} \boldsymbol{G}_{ij} \tag{4.2}$$

  where $\boldsymbol{G}_i$ is the gradient image at direction i.

- Finally, to compute the final gradient image of the whole HSI cube, the average of the four directional image gradients is computed

$$\boldsymbol{G} = (1/4) \times \sum_{i=1}^{4} \boldsymbol{G}_i \tag{4.3}$$

where $\boldsymbol{G}$ is the output gradient image of the input HSI.

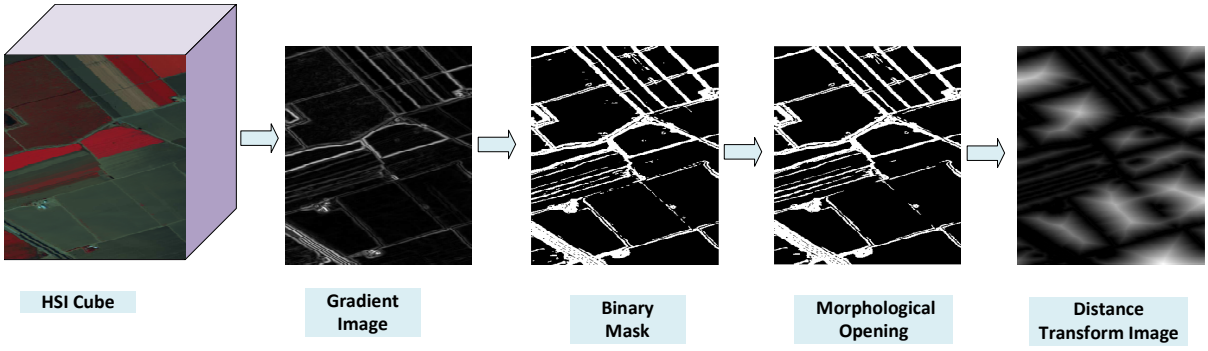


Figure 4.2: Steps of obtaining the distance transform image of the Salinas hyperspectral dataset.

Having calculated the gradient image in order to extract strong edges, we thresholded the gradient image. To remove small connected components in the thresholded image, morphological opening is applied on the resulting image. Finally, in order to find the distance of the pixels in the original image to the foreground (edge) pixels, we used distance transform [87] with euclidean distance metric

$$D_{euc} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.4}$$

where (x_1, y_1) and (x_2, y_2) represent two pixels' locations in the image. In the resulting distance transform image, the darker the pixel is, it is closer to an edge and vice versa. Figure 4.2 shows the steps of obtaining the distance transform image of the Salinas hyperspectral dataset as an example.

Having calculated the distance transform values, we included them in the spatial feature vector as is shown in the block diagram of our proposed method in Figure 4.3. In order to find the spatial features of the pixels in the HSI, we first reduced the dimension of the data along its spectral axis using the PCA method. For each target pixel (blue square), a surrounding neighborhood area (green region) is considered. Then, to form our

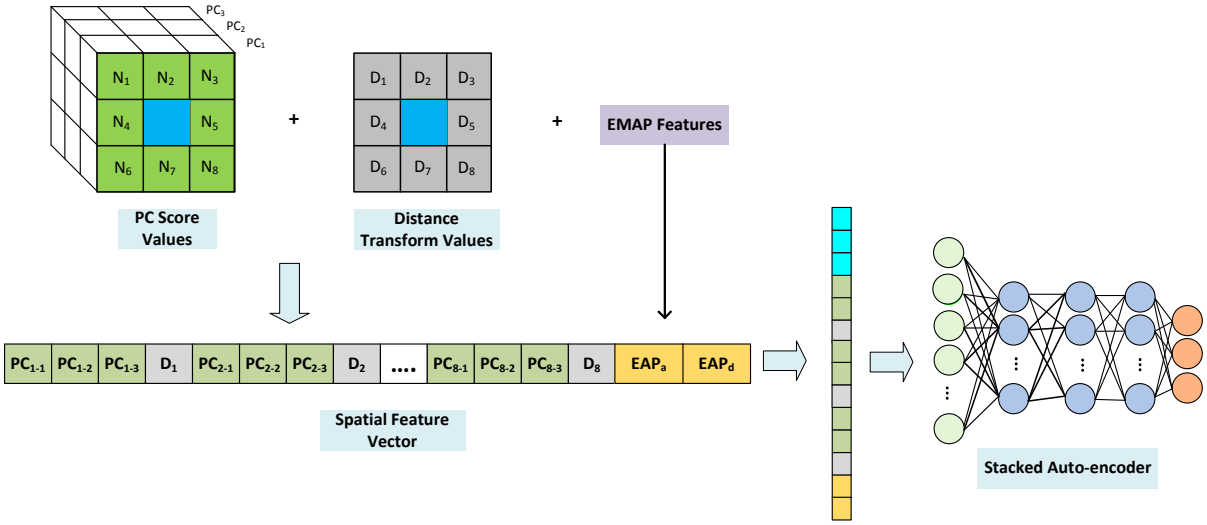


Figure 4.3: Block diagram of our proposed method. The cube shown in the top row depicts only a neighborhood region around the blue pixel. Also, the spectral dimensionality of the input HSI (not shown in this figure) is reduced using the PCA method and as an example 3 PCs are retained.

primary spatial feature vector, PC values of the neighbors and distance transform values from the corresponding distance transform image (Figure 4.2) are combined according to (4.5)

$$\begin{aligned} \boldsymbol{S}_{\boldsymbol{z}p} &= horzcat(\boldsymbol{PC}_i, D_{\boldsymbol{x}_i}) \\ s.t \quad & i = 1, ..., p \end{aligned} \tag{4.5}$$

where $\boldsymbol{S}_{\boldsymbol{z}p}$ and $\boldsymbol{x}_i$ represent the primary spatial feature vector associated with the target pixel $\boldsymbol{z}$ and the ith pixel in the neighborhood region around this pixel, respectively. $\boldsymbol{PC}_i$ and $D_{\boldsymbol{x}_i}$ represent a vector containing the PC score values and the distance transform value (scalar) associated with pixel $\boldsymbol{x}_i$, respectively. Also, $horzcat(.)$ indicates horizontal concatenation. The closer the neighbor is to an edge, the lower its corresponding distance value is; so, we want that neighbor to have less contribution in adding spatial information to the target pixel. In other words, we do not want all neighbors to contribute the same in adding spatial information to the target pixel.

To extract even more spatial information by adding geometric attributes to the primary spatial feature vector similar to [48], we incorporated EMAP features (equation

4.7) to $\boldsymbol{S_{zp}}$ and formed our *secondary* spatial feature vector

$$\boldsymbol{S_{zs}} = [\boldsymbol{S_{zp}}, \quad EMAP] \tag{4.6}$$

where

$$EMAP = \left\{ EAP_{a_1}, EAP'_{a_2}, ..., EAP'_{a_n} \right\} \tag{4.7}$$

In equation (4.7), each EAP is computed according to (4.8). (For a full explanation of the structure of EMAP features please refer to Section 2.3.5)

$$EAP = \{AP(PC_1), AP(PC_2), ..., AP(PC_k)\} \tag{4.8}$$

Finally, the pixel's spectrum is added to the proposed spatial feature vector to form the input data to the stacked autoencoder.

In Section 4.3, we carry out extensive experiments to show the effectiveness of our proposed method in classification of remote sensing hyperspectral scenes. In fact, we performed three sets of experiments using different kinds of spatial feature vectors combined with the spectrum of each sample. We itemized the three proposed spatial feature vectors in this chapter as follows:

- First, we used PCA dimensionality reduction method and our primary proposed feature vector, called *Proposed-P* (the one without the EMAP features).
- Next set of experiments was done using PCA and our second feature vector including EMAP features, called *Proposed-S*.
- Finally, we used PCDA [83] DR method to reduce the spectral dimension of the input HSI and combined it with our primary feature vector to form a new type of spatial features. Process of forming this final type of spatial feature vector and its employment in the classification problem is shown in Figure 4.4. This set of experiments was performed to investigate the effect of using class labels in the DR

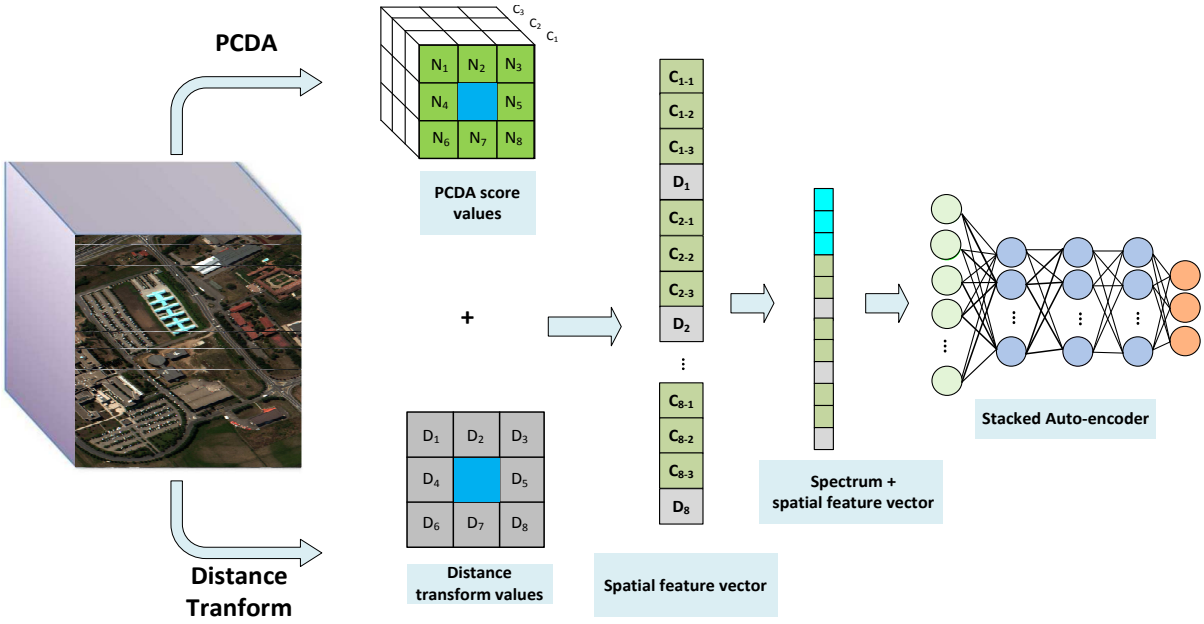


Figure 4.4: Block diagram of the proposed feature vector obtained by PCDA and distance transform values.

step prior to form our first proposed spatial feature vector on classification results.

4.3 Experimental Results

To check the effectiveness of our proposed method we applied it on three hyperspectral datasets listed in Section 4.3.1. The metrics used for the performance evaluation are class-specific accuracy, overall accuracy (OA), average accuracy (AA), and Kappa coefficient. In remote sensing HSI applications, there is normally a limited number of available ground truth pixels. Consequently, training a deep learning model becomes problematic due to the need of many ground truth samples to train a deep neural network whose architecture has many trainable parameters. As a result, it is important to check the reliability of a deep learning-based classifier versus using only a small portion of available labeled pixels for training. Therefore, in the experiments we used only 10% of the labeled samples for training and the rest 90% for testing. We repeated each experiment 10 times and reported the mean value along with standard deviation of the accuracy metrics. It should

be noted that in each of these 10 iterations, training samples are picked randomly.

4.3.1 HYPERSPECTRAL DATASETS

4.3.1.1 Salinas

This dataset was collected by the means of AVIRIS in 1998 over Salinas Valley, California. Salinas hyperspectral dataset originally included 224 spectral bands. However, having removed the 20 water absorption bands, it is left with 204 bands. This database contains an image of size 512 $\times$ 217 pixels in each band and has the high spatial resolution of 3.7 meters. The ground truth of the Salinas scene covers 16 classes including vegetables, bare soil, and vineyard fields. Figure 4.5 shows the false color image of this dataset along with its pseudo color ground truth. Number of available labeled, train, and test samples used in this chapter are listed in Table 4.1.

4.3.1.2 University of Pavia

This database has been gathered by a ROSIS over the University of Pavia, northern Italy. University of Pavia dataset has 103 spectral bands in the wavelength range of 0.43-0.86 μm. The spatial extent of each band is 610$\times$340 pixels. The ground truth image contains 9 different classes. False color image of this dataset and its corresponding pseudo color ground truth are shown in Figure 4.6. Number of available samples for the University of Pavia dataset and the number of train and test samples used in this chapter are listed in Table 4.2.

4.3.1.3 Surrey

This dataset is a small subscene of the hyperspectral image captured by the airborne CASI-1500 sensor over the city of Surrey, BC, Canada in April 2013. This HSI includes

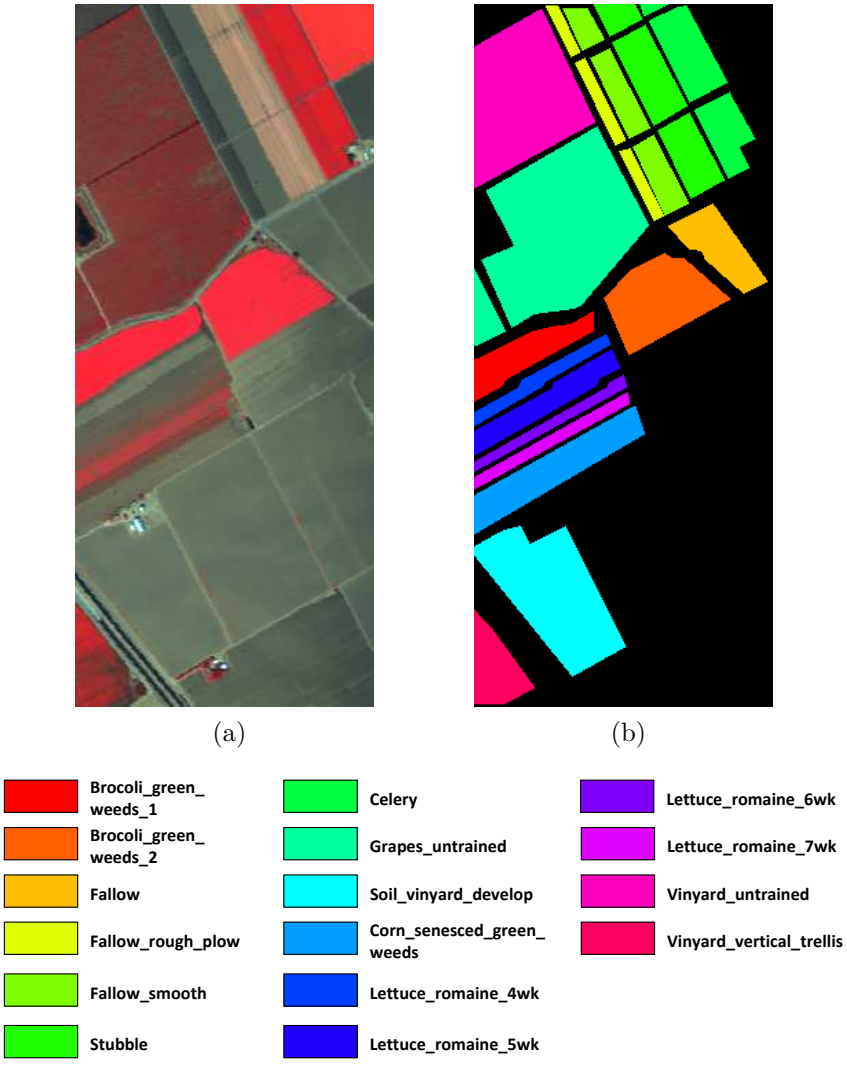


Figure 4.5: Salinas dataset. (a) False color image. (b) Pseudo ground truth image.

72 spectral bands in the range of 0.36-1.05μm with the spectral resolution of 9.6 nm and the high spatial resolution of 1 m. The available ground truth includes 5 different classes. The false color and the pseudo color ground truth images are shown in Figure 4.7. Table 4.3 lists number of available samples, train and test ratios used in this chapter for the Surrey dataset.

Table 4.1: Number of labeled samples for the different sixteen classes of the Salinas dataset along with the number of train and test samples used in this chapter.

No	Class	Num. of samples	Train	Test
1	Brocoli-green-weeds-1	2009	201	1808
2	Brocoli-green-weeds-2	3726	373	3353
3	Fallow	1976	198	1778
4	Fallow-rough-plow	1394	139	1255
5	Fallow smooth	2678	268	2410
6	Stubble	3959	396	3563
7	Celery	3579	358	3221
8	Grapes-untrained	11271	1127	10144
9	Soil vineyard develop	6203	620	5583
10	Corn-senesced-green-weeds	3278	328	2950
11	Lettuce-romaine-4wk	1068	107	961
12	Lettuce-romaine-5wk	1927	193	1734
13	Lettuce-romaine-6wk	916	92	824
14	Lettuce-romaine-7wk	1070	107	963
15	Vineyard-untrained	7268	727	6541
16	Vineyard-vertical-trellis	1807	181	1626
	Total	54129	5415	48714

4.3.2 Parameter Tuning

Even though network's parameters are tuned during the training step, there are hyper-parameters in the model that need to be carefully set. These hyperparameters include the number of retained PCs during the dimensionality reduction step, n, size of the neighborhood region around each target pixel, s, number of the neurons in each layer of the network, and the required threshold parameters in the distance transform image acquisition process, T_1 and T_2.

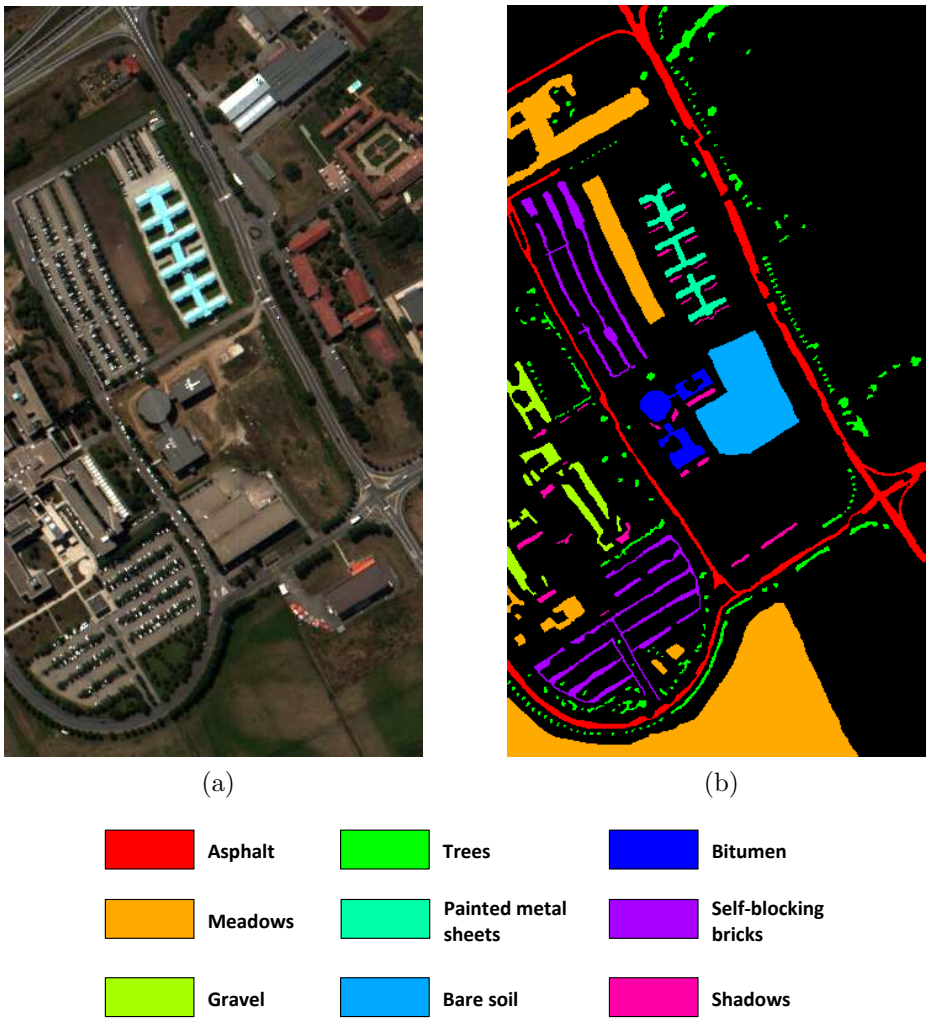


Figure 4.6: University of Pavia dataset. (a) True color image. (b) Pseudo ground truth image.

4.3.2.1 Number of retained PCs and size of the neighborhood

In this set of experiments, we used our primary spatial feature vector and tried to find the best values for n and s hyperparameters. We considered keeping 1 to 10 PCs during the dimensionality reduction step. For the neighborhood size, we examined the following window sizes: 3×3, 5×5, 7×7, and 9×9. Distribution of the OA these two hyperparameters are depicted in Figure 4.8 for the three hyperspectral datasets.

As Figure 4.8a shows there is a general increase in the OA versus n especially for $s = 3, 5$, and 7. Increasing the size of the neighborhood area has a positive effect on the OA

Table 4.2: Number of labeled samples for the different nine classes of the University of Pavia dataset along with the number of train and test samples used in this chapter.

No	Class	Num. of samples	Train	Test
1	Asphalt	6631	663	5968
2	Meadows	18649	1865	16784
3	Gravel	2099	210	1889
4	Tress	3064	306	2758
5	Painted metal sheets	1345	135	1210
6	Bare Soil	5029	503	4526
7	Bitumen	1330	133	1197
8	Self-Blocking Bricks	3682	368	3314
9	Shadows	947	95	852
	Total	42776	4278	38498

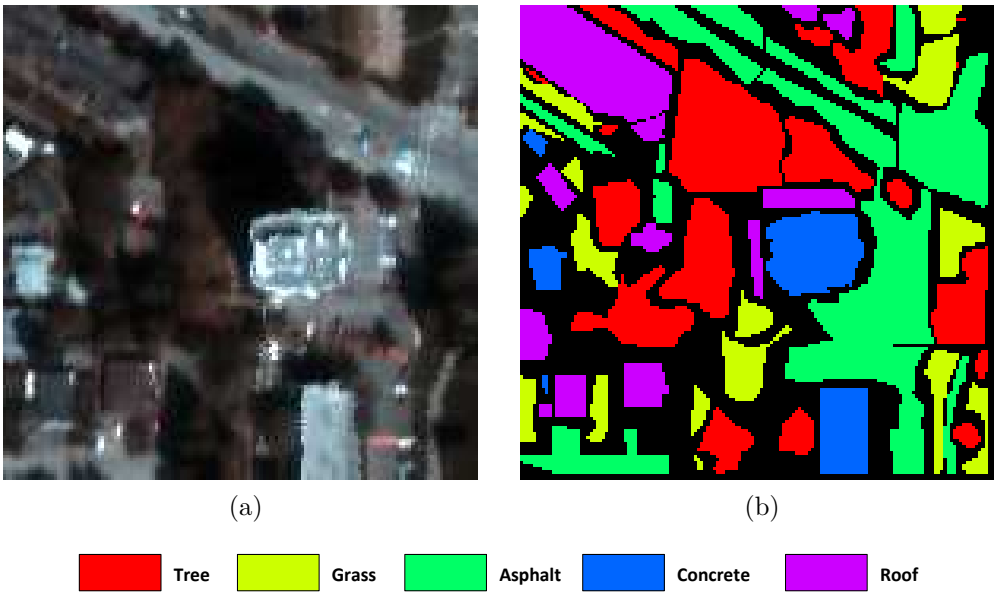
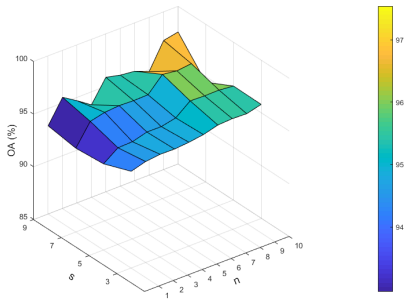


Figure 4.7: Surrey dataset. (a) False color image. (b) Pseudo ground truth image.

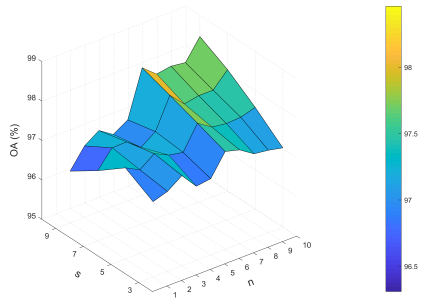
as well. However, in the case of $s = 9$, for some values of n OA drops compared to some of its corresponding values for $s = 7$. To have a trade off between the model complexity and the accuracy, for n and s we chose values of 5 and 7, respectively. According to Figure 4.8b, OA generally increases with the increase of n. However, to have a trade off between the accuracy and the complexity of the feature vector, the value of 5 was chosen for this hyperparameter. Also, neighborhood window is chosen to be 7×7 since it results in the best OA when n is set to 5.

Table 4.3: Number of labeled samples for the different five classes of the Surrey dataset along with the number of train and test samples used in this chapter.

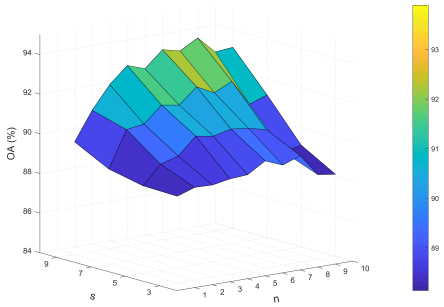
No	Class	Num. of samples	Train	Test
1	Tree	4121	412	3709
2	Grass	1847	185	1662
3	Asphalt	4375	438	3937
4	Concrete	1241	124	1117
5	Roof	2151	215	1936
	Total	13735	1374	12361



(a)



(b)



(c)

Figure 4.8: OA obtained by our primary spatial feature vector (Proposed-P) vs n and s for (a) Salinas, (b) University of Pavia, and (c) Surrey datasets.

For the Surrey dataset, we observed a general increase in the OA with the size of the neighborhood (Figure 4.8c). Increasing the value of parameter n results in a generally higher OA up to $n = 8$. To have an agreement between the accuracy and the size of the feature vector, similar to the case of University of Pavia, parameters s and n are chosen

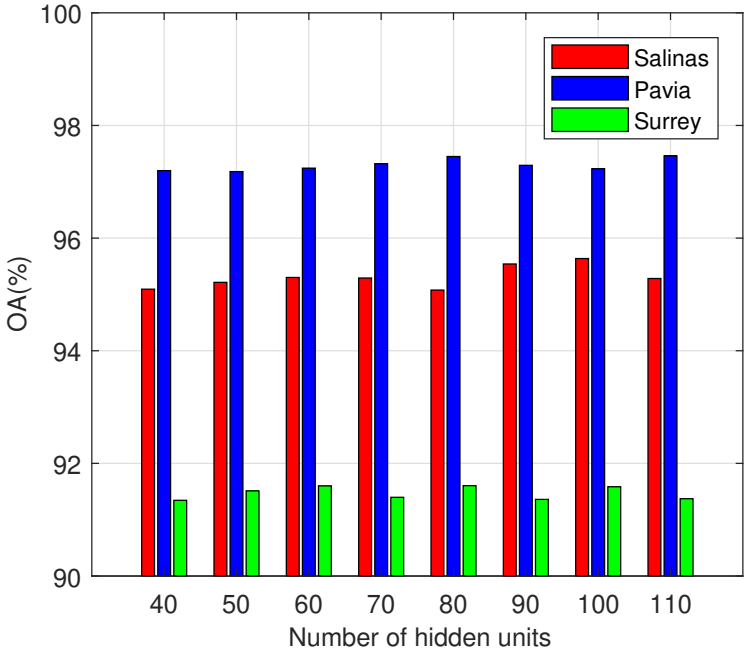


Figure 4.9: OA versus number of hidden units in each layer for the three hyperspectral datasets.

to be 7 and 5, respectively.

4.3.2.2 Size of the hidden layers

One of the hyperparameters that need to be set, is number of neurons in each layer. We tried eight values for this parameter for the three databases. Figure 4.9 shows the OA versus the size of the hidden layers for the three HSI databases. As can be seen from this figure, number of hidden units does not have much effect on th OA. So, in order to have an optimized number of trainable parameters in the network we chose the value of 60 for this hyperparameter for the three datasets.

4.3.2.3 Required threshold parameters

In our method, there are two thresholds in the process of obtaining the distance transform image of each dataset: A threshold above which pixels in the gradient image are con-

sidered strong edges T_1, and a threshold for removing the connected components having fewer than P pixels in the binary mask image (see Figure 4.2) T_2. We tried five different values for T_1 and T_2 for all three datasets in this study. Figure 4.10 shows the distribution of the OA versus the five values for these two parameters. For the Salinas dataset, T_1 and T_2 equal to 0.08 and 14 gave the highest OA. For the University of Pavia dataset, values of 0.31 and 50 for the T_1 and T_2 resulted in the highest OA. Also, for the Surrey dataset, T_1 and T_2 equal to 0.2 and 28 led to the best OA. Class specific accuracies, OA, AA, and the kappa coefficient corresponding to these values are listed in the second last column of Tables 4.4-4.6.

4.3.3 Performance Evaluation

4.3.3.1 Effect of using supervised dimensionality reduction

To see whether using class labels in the DR step has any effect on the classification results, we used PCDA method to reduce the spectral dimension of the input HSI datasets as shown in Figure 4.4. This block digram is similar to the one shown in Figure 4.3. However, PCDA is employed instead of PCA. Also, here, we did not incorporate EMAP features in the spatial feature vector. In this experiments, for parameters s, T_1, T_2, and *size of the hidden layers* we used the optimal values that have been obtained in the parameter tuning step presented in Section 4.3.2. As was mentioned in Section 3.3.2, PCDA method includes two parameters, n_1 and n_2. In order to be able to have a fair comparison between the results of this section and the ones obtained using the first type of features mentioned at the end of Section 4.2, we chose n_1 and n_2 such that their total equals to the value of parameter n chosen in Section 4.3.2.1. Since we observed that for a fixed value of n, $n_2 > n_1$ results in better accuracies, the values of $n_1 = 1$ and $n_2 = 4$ have been chosen for the three HSI datasets. Results of applying the method described in this subsection are shown in Table 4.7 for the three datasets. Comparing accuracies shown in this table to the result obtained from our primary feature vector shows an improvement in the reported accuracy values and proves the benefit of using a supervised DR method

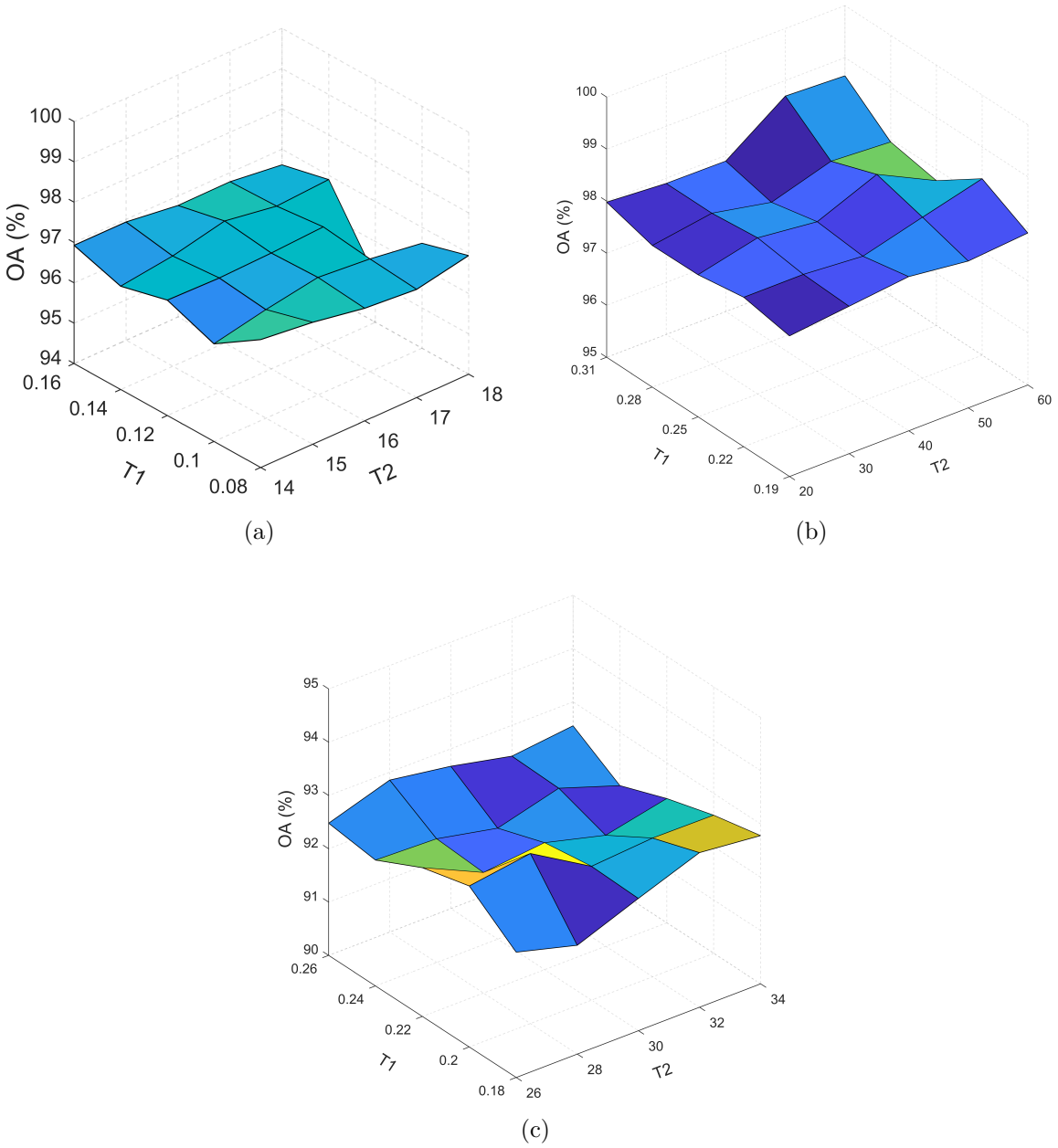


Figure 4.10: OA obtained by the Proposed-P feature vector vs parameters T_1 and T_2 for (a) Salinas, (b) University of Pavia, and (c) Surrey datasets.

prior to form the proposed spatial feature vector.

Table 4.4: Classification accuracies (%) and test time (s) of different methods for Salinas dataset using 10% of the training data.

No	L-SVM	K-SVM	EMAP	DAE	PPF-CNN	EMAP-SAE	Proposed-P	Proposed-S
1	99.39	99.60	100	99.61	99.17	99.87	100	99.90
2	99.70	99.69	99.95	99.70	99.66	99.70	99.94	99.85
3	99.58	99.63	99.84	98.49	98.38	99.35	100	99.30
4	99.44	99.39	15.68	99.02	99.28	98.88	98.88	99.58
5	98.87	98.93	98.40	98.01	98.56	98.76	99.46	98.68
6	99.88	99.74	99.84	99.75	99.83	99.72	99.99	99.94
7	99.75	99.58	99.86	99.24	99.81	99.47	99.86	99.63
8	88.40	88.39	87.92	91.17	94.58	89.72	93.34	95.46
9	99.81	99.80	99.84	99.74	99.82	99.73	99.91	99.83
10	97.48	97.37	96.68	97.09	96.28	96.51	98.59	98.47
11	98.71	98.93	97.63	97.13	96.34	97.61	97.92	99.17
12	99.79	99.92	94.51	99.92	99.81	99.44	99.85	100
13	99.11	99.61	29.70	99.42	98.54	97.63	99.27	99.94
14	97.32	97.79	95.12	98.68	96.36	95.53	98.18	99.47
15	65.03	74.94	62.74	88.17	82.12	82.87	91.25	94.00
16	98.73	98.71	99.83	97.79	97.79	99.25	99.75	98.86
OA	92.42 ±0.12	93.75 ±0.13	88.45 ±0.19	95.92 ±0.22	95.75 ± 0.69	94.90 ±0.12	97.17 ±0.17	**97.93 ±0.15**
AA	96.31 ±0.11	97.00 ±0.08	86.10 ±0.12	97.68 ± 0.22	97.27 ±0.46	97.13 ±0.21	98.51 ±0.11	**98.88 ±0.08**
Kappa	0.91 ±0.001	0.93 ± 0.001	0.87 ±0.002	0.95 ±0.002	0.95 ±0.008	0.94 ±0.001	0.97 ±0.002	**0.98 ±0.002**
Test time (s)	14.10	12.32	7.03	0.48	10.36	0.59	0.08	0.09

4.3.3.2 Comparison with other methods

In order to check the effectiveness of our proposed method, we compared it with some traditional and recent hyperspectral image classification methods. These methods include linear SVM, RBF SVM, EMAP, DAE [41], PPF-CNN [47], and EMAP-SAE [48].

For the linear and RBF SVM, we performed 10-fold cross validation to find the best values for the regularization parameter C and the width of the gaussian kernel *gamma*. For the Salinas database, value of 10^3 for C in case of linear SVM and values of 10^3 and 1 for C and *gamma* parameters of the gaussian SVM resulted in the best performance. For the University of Pavia dataset, C equal to 10 for the linear SVM and in case of the RBF SVM values of 10^3 and 0.1 for C and *gamma* outperformed the other values. For the

Table 4.5: Classification accuracies (%) and test time (s) of different methods for University of Pavia dataset using 10% of the training data.

No	L-SVM	K-SVM	EMAP	DAE	PPF-CNN	EMAP-SAE	Proposed-P	Proposed-S
1	85.04	89.17	90.08	94.45	98.82	98.20	97.29	99.29
2	95.11	96.32	97.85	98.68	99.28	99.31	99.86	99.87
3	68.05	71.97	93.75	86.26	82.30	94.13	96.87	99.09
4	89.93	92.32	96.36	95.73	93.22	95.74	97.72	97.69
5	99.62	99.47	99.31	99.64	99.73	99.02	99.83	99.61
6	55.31	72.67	91.07	89.94	95.42	97.86	99.05	98.99
7	72.79	75.23	89.66	85.03	87.05	98.43	97.31	99.72
8	72.06	75.98	94.51	90.68	92.58	95.99	96.61	98.52
9	93.84	94.84	77.89	99.21	99.42	99.35	99.39	99.17
OA	84.61 ±0.23	88.61 ± 0.17	94.60 ±0.07	95.11 ±0.46	96.55 ±0.32	98.13 ±0.44	98.70 ±0.10	**99.34 ±0.11**
AA	81.31 ± 0.35	85.33 ±0.45	92.28 ±0.26	93.29 ± 0.62	94.20 ±1.10	97.56 ±0.46	98.21 ±0.12	**99.11 ±0.18**
Kappa	0.79 ±0.003	0.85 ±0.002	0.93 ±0.001	0.93 ±0.006	0.95 ±0.004	0.97 ±0.004	0.98 ±0.001	**0.99 ±0.001**
Test time (s)	11.01	10.02	4.07	0.25	7.03	0.35	0.06	0.07

Table 4.6: Classification accuracies (%) and test time (s) of different methods for Surrey dataset using 10% of the training data.

No	L-SVM	K-SVM	EMAP	DAE	PPF-CNN	EMAP-SAE	Proposed-P	Proposed-S
1	88.60	90.06	88.42	92.95	95.11	91.69	95.25	94.15
2	68.40	73.11	83.92	83.69	75.92	82.66	90.55	88.58
3	88.80	89.57	93.50	90.65	92.50	93.65	93.76	96.19
4	91.54	91.20	98.38	90.68	96.72	92.45	94.28	96.49
5	76.70	76.86	87.50	84.54	86.46	91.39	89.74	94.48
OA	84.35 ±0.43	85.66 ±0.49	90.19 ± 0.46	89.45 ±0.54	90.49 ±0.47	91.12 ±0.63	93.19 ±0.27	**94.31 ±0.25**
AA	82.81 ± 0.65	84.16 ±0.81	90.34 ±0.52	88.51 ±0.66	89.34 ±0.59	90.37 ±0.75	92.72 ±0.35	**93.98 ±0.24**
Kappa	0.79 ±0.006	0.81 ±0.007	0.87 ±0.006	0.86 ± 0.007	0.87 ±0.006	0.88 ±0.008	0.91 ±0.003	**0.92 ±0.003**
Test time (s)	9.03	7.12	3.81	0.18	6.91	0.24	0.05	0.06

Surrey dataset, these values have been obtained as 10^3, 10^5, and 0.01, respectively. We used the RBF SVM classifier with the EMAP method and found the fowling values for the gaussian SVM's parameters for the three datasets: Salinas: C=10^6 and $gamma$=0.1, University of Pavia: C=10^4 and $gamma$=0.001, and Surrey: C=10^3 and $gamma$=0.01. With all other methods we used logistic regression (LR) classifier with softmax activation function.

Table 4.7: Classification accuracies (%) obtained from the last set of experiment, using PCDA dimensionality reduction method and our primary proposed feature vector, for the three HSI datasets using 10% of the training data.

No	Salinas	Pavia university	Surrey
1	99.80	98.78	95.21
2	99.94	99.94	89.36
3	99.80	99.27	95.71
4	99.16	98.62	97.84
5	99.54	99.40	92.27
6	99.96	99.66	-
7	99.78	97.99	-
8	95.81	98.25	-
9	99.91	99.46	-
10	98.99	-	-
11	99.17	-	-
12	99.99	-	-
13	99.30	-	-
14	99.15	-	-
15	93.56	-	-
16	99.38	-	-
OA	98.04±0.09	99.37±0.07	94.36±0.14
AA	98.95 ±0.10	99.04± 0.14	94.08±0.27
Kappa	0.98± 0.001	0.99 ±0.0	0.92±0.004

Tables 4.4-4.6 compare the result of our proposed method with the other methods experimented in this study. As can be seen from these tables, adding the distance transform value to the feature vector and combining it with EMAP features improves the accuracy metrics. For the Salinas dataset in terms of OA, adding distance transform features improves the result by 2.27% compared to the EMAP-SAE method. For the University of Pavia and Surrey datasets, the corresponding increase in the OA is 0.57% and 2.07%, respectively. Also, the increase in the AA compared to the EMAP-SAE method for the three datasets have been observed as 1.38 %, 0.65%, and 2.35%, respectively.

The last column of these tables lists our results using our secondary proposed feature vector. For the Salinas database, the OA, AA, and the kappa coefficient increased by 0.76%, 0.37%, and 1%, respectively compared with employing only the primary spatial feature vector. The corresponding values for the second database are 0.64%, 0.9%, and 1%. Finally, for the Surrey dataset, these accuracy metrics increased by 1.12%, 1.26%, and 1%, respectively.

Figures 4.11-4.13 show the classification maps obtained from the different methods explored in this study. As can be seen from these maps, there are fewer misclassified pixels in our results which is consistent with the accuracies shown in Tables 4.4-4.6. For instance, according to Figure 4.11 (i) (and also Table 4.4), our proposed secondary spatial feature vector especially improves the classification accuracies of classes 8 and 15 of the Salinas dataset. As Figure 4.12 (i) shows, the proposed-S feature vector improves the classification accuracy of class 3 of the University of Pavia dataset significantly compared to other methods.

4.4 Conclusion

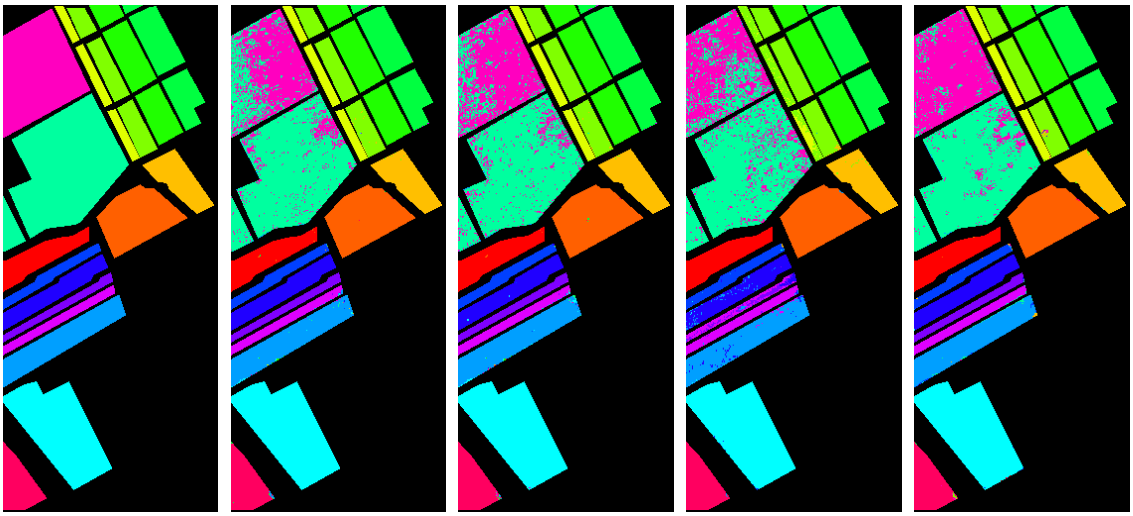
In this chapter, we proposed a new method for the HSI classification problem. In an HSI, pixels inside a neighborhood have the same class label with a high probability. Therefore, it seems reasonable to use information of the neighbors of a target pixel as its spatial features. However, the idea behind our method is that spatial location of the pixel in the HSI matters and not all neighbors should contribute the same in forming target pixel's spatial feature vector. To accomplish this, we introduced a new distance transform-based spatial feature vector which considers the distance of pixels with respect to the edges in the image as new features. To find such distance values we proposed a method to compute the distance transform image of an input hyperspectral image.

We employed a two-layer stacked autoencoder consisting of a sparse autoencoder in each layer as the deep learning framework. In order to find the best values for param-

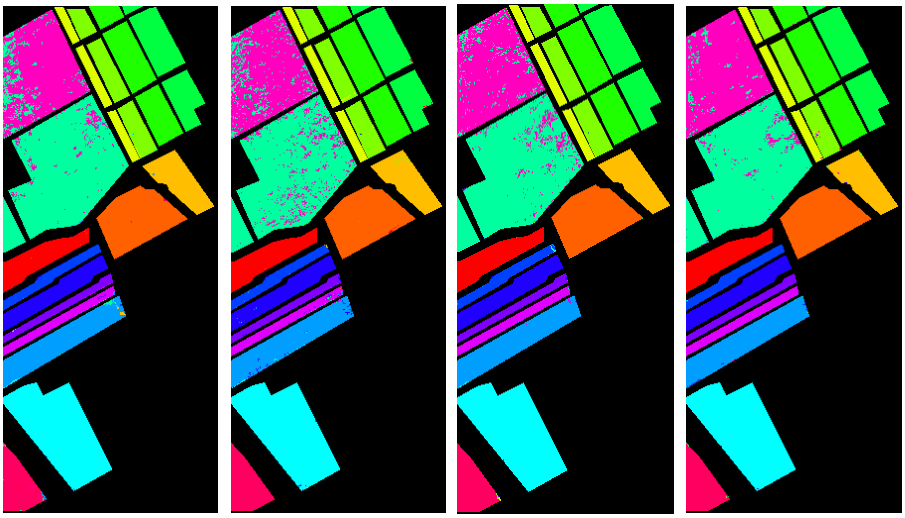
eters of our method, we performed extensive experiments in the hyperparameter space. Furthermore, to add more spatial information, we incorporated EMAP features to our proposed spatial feature vector as well. Finally, to examine whether using class labels in the DR step has any effect on the classification results, we used PCDA method to reduce the spectral dimension of the input HSI datasets. To evaluate the effectiveness of our proposed method, we applied it on three HSI datasets, Salinas, University of Pavia, and Surrey and performed three sets of experiments.

In the first set, we employed PCA to reduce the spectral dimension of the HSI and used our proposed distance transform-based spatial feature vector. In the second set, we add EMAP features as well. Third set includes using PCDA and distance transform-based spatial feature vector. According to the results of the first set, including distance features improved the classification accuracies proving the effectiveness of our proposed approach in contributing the neighbors spatial information with different weights. The improvements in the classification accuracies obtained from the second set (compared to the first set) demonstrates the power of EMAP features in modeling spatial information and the benefit of using them in our proposed spatial feature vector. Results of the third set revealed that although using PCDA and the proposed distance transform features improve the classification accuracies compared to the first set, they are quite close the results of the second set.

Comparing classification results of these three sets with some conventional methods and some recent deep learning-based approaches shows the superiority of our approach in classification of remote sensing hyperspectral scenes.



(a) (b) (c) (d) (e)



(f) (g) (h) (i)

Figure 4.11: Salinas (a) ground truth, (b)-(i) classification maps resulting from different methods. (b) Linear SVM, (c) kernel SVM, (d) EMAP, (e) DAE, (f) PPF-CNN, (g) EMAP-SAE, (h) Proposed-P, and (i) Proposed-S

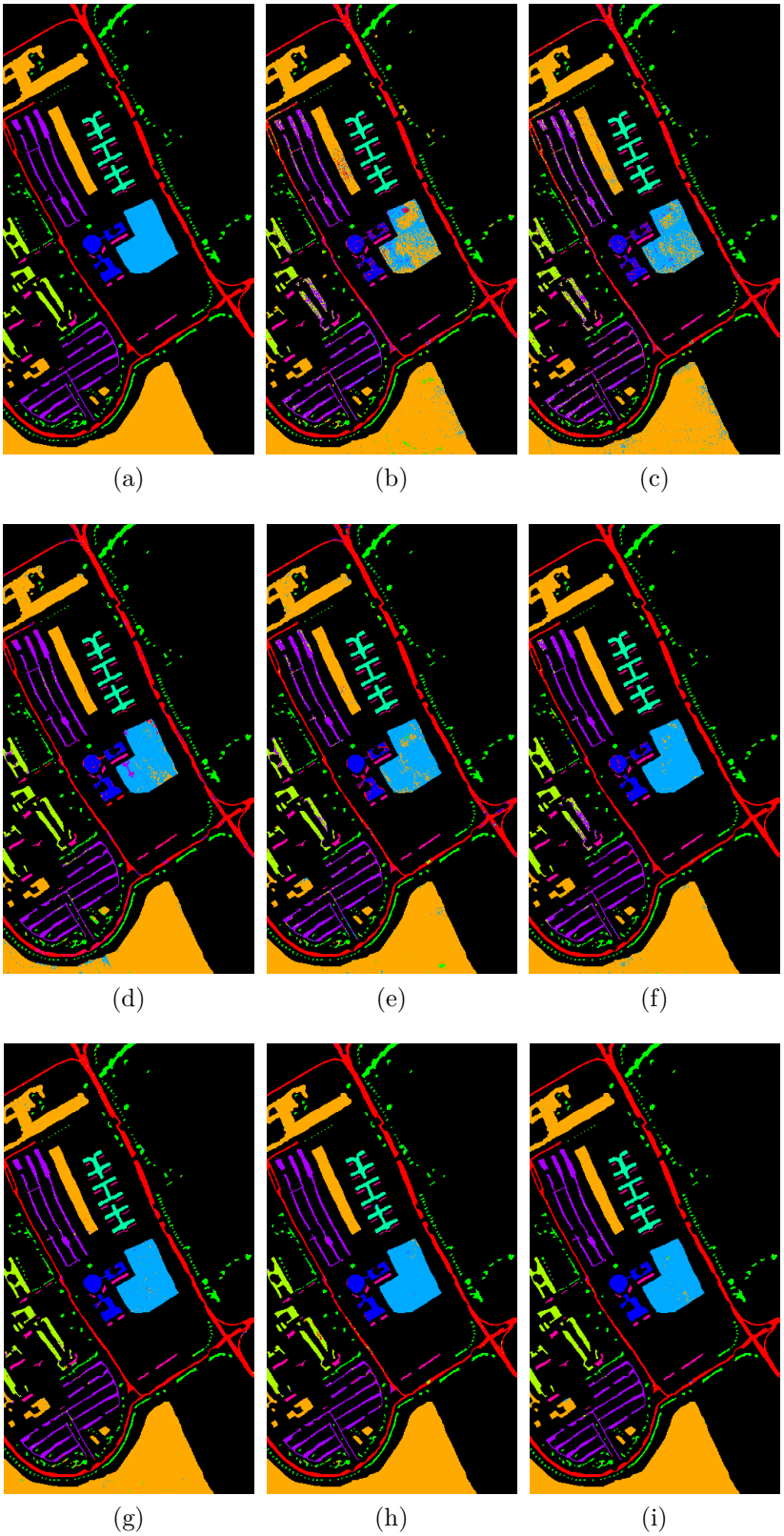


Figure 4.12: University of Pavia (a) ground truth, (b)-(i) classification maps resulting from different methods. (b) Linear SVM, (c) kernel SVM, (d) EMAP, (e) DAE, (f) PPF-CNN, (g) EMAP-SAE, (h) Proposed-P, and (i) Proposed-S

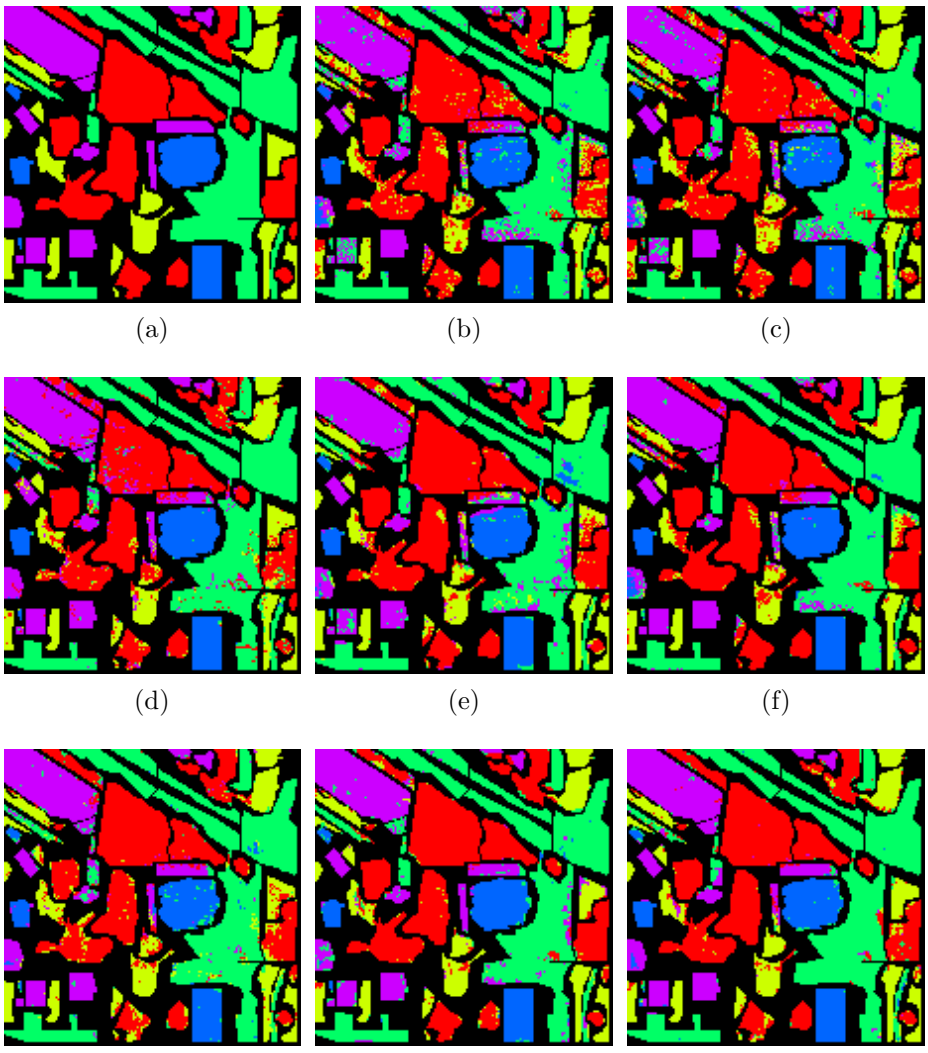


Figure 4.13: Surrey. (a) ground truth, (b)-(i) classification maps resulting from different methods. (b) Linear SVM, (c) kernel SVM, (d) EMAP, (e) DAE, (f) PPF-CNN, (g) EMAP-SAE, (h) Proposed-P, and (i) Proposed-S

Chapter 5

Spectral perturbation method for deep learning-based classification of remote sensing hyperspectral images

5.1 Introduction

Deep learning-based hyperspectral image classification has become very popular recently. It has been widely used in the domain of classification of remote sensing hyperspectral images and has shown encouraging results. Although presence of many spectral bands in a hyperspectral image provides valuable features for the classification purposes, lack of adequate training samples makes training a deep learning model a challenging task. One solution that has been used in the literature to mitigate this problem is data augmentation [47, 54].

In this chapter, we employed stacked auto-encoder (SAE) as the deep learning architecture to extract deep spectral features of the input data and to address the problem of absence of enough training samples, we propose a simple yet effective data augmentation approach to boost the number of training data. Also, in order to alleviate the noise in

the output image, we used the majority voting strategy to smooth the final classification map. We applied our method on the Indian Pines hyperspectral dataset including very few training examples. Experimental results show the superiority of our method compared to some conventional and recent HSI classification methods.

The rest of this chapter is organized as follows: Section 5.2 presents our proposed algorithm. Database description and the experimental results are presented in Section 5.3 and Section 5.4 concludes this chapter.

5.2 Method

In [47], original data is augmented to train a CNN and in the test phase majority voting is applied on the output to smooth the primary classification map. Similarly, in this chapter to overcome the problem of insufficiency of training samples to train our SAE, we propose a simple data augmentation method. Also, in the test step to incorporate the spatial information, we smoothed the classification map using the information of the neighboring pixels of each test pixel by the means of majority voting technique. Our proposed algorithm consists of the following steps:

First, the available labeled samples are divided into two sets of train and test. Then training samples are augmented using our proposed data augmentation technique. Next, in the training phase, the original and augmented training data are used to train a SAE network. In the test phase, test samples are given to the trained SAE and an initial classification map is produced. So far we have only used spectral information. So, in order to incorporate the spatial information, similar to the algorithm in [47], we used majority voting strategy to smooth the output classification map at the test phase.

Spectra of some of the classes in the Indian Pines dataset is shown in Figure. 5.1. As can be seen from this figure (and also Table. 5.1), some of the classes have very few training samples. The idea behind our augmentation approach is that any spectrum with

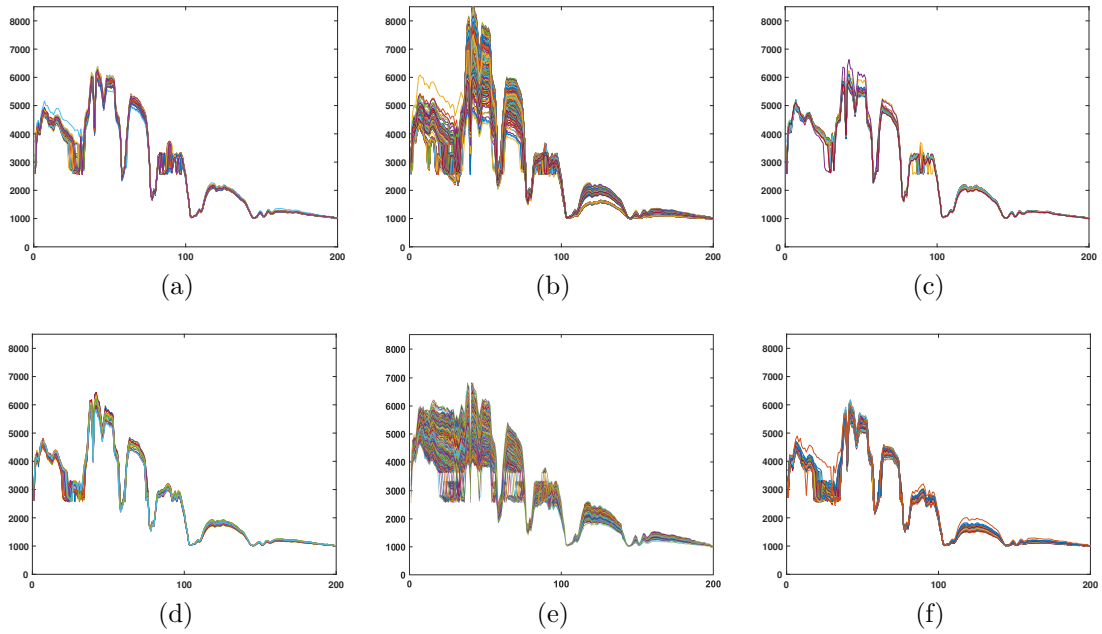


Figure 5.1: Spectra of some of the classes in the Indian Pines dataset. (a) Alfalfa, (b) Grass-pasture, (c) Grass-pasture-mowed, (d) Oats, (e) Soybean-clean, and (f) Wheat.

the similar pattern to the samples of class k that falls within the reflectance range covered by these samples can be considered as an imaginary sample of this class as well. That being said, we created n samples out of each original sample in each class. Each new sample is created by shifting the original spectrum up or down within a specific range (*shift range*) such that the generated spectra fall within the reflectance range overlaid by the original samples. Process of generating n virtual spectra of the ith original sample of class k, $\boldsymbol{x}_i$, can be formulated as follows:

$$\boldsymbol{x}_{i,j} = \boldsymbol{x}_i + \boldsymbol{R}_j \quad , \quad j = 1, 2, ..., n. \tag{5.1}$$

where $\boldsymbol{R}_j$ is a vector with the same size as $\boldsymbol{x}_i$ and its elements are a random integer drawn from the discrete uniform distribution on the interval $[-S \quad +S]$ where S indicates the desired shift range in the reflectance domain. As an example, Figure 5.2 shows the original and the augmented spectra of class *Alfalfa* with n and *shift range* equal to 100 and 150, respectively. In other words, we created 100 new spectra of each original sample of this class by linearly shifting the input up or down within the reflectance range of -150

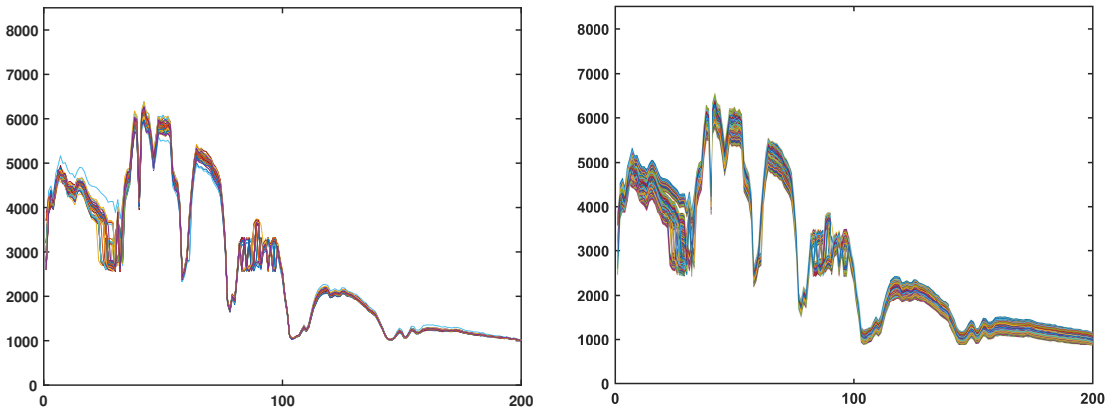


Figure 5.2: (a) Original spectra of class Alfalfa of the Indian Pines dataset and (b) augmented spectra of the same class.

and 150.

In order to smooth the classification map, we performed *majority voting* strategy [47]. Since in a remote sensing HSI, neighboring pixels belong to the same class with a high probability, after obtaining the classification map, for each test pixel, we consider a $k \times k$ neighborhood area around it and assign the test pixel to the class the majority of its neighbors belong to. This process can be formulated as (5.2)

$$L\left(\boldsymbol{x}_i\right) = Mode\left(L(\boldsymbol{x}_j)|\boldsymbol{x}_j \in N_{\boldsymbol{x}_i}\right), \tag{5.2}$$

where L stands for label, $\boldsymbol{x}_i$ and $N_{\boldsymbol{x}_i}$ indicate the ith test sample and its neighborhood area, respectively.

5.3 Experimental Results

In this section, we evaluate the performance of our proposed method on the well-known online hyperspectral database, Indian Pines. In the experiments, we used 20% of the labeled samples for training and the remaining 80% for testing except for the classes with less than 100 samples where we used half of the data for training and the rest

Table 5.1: Number of labeled samples, train, and test pixels for the sixteen classes in the Indian Pines dateset.

No	Class	Available data	Train	Test
1	Alfalfa	46	23	23
2	Corn-notill	1428	286	1142
3	Corn-mintill	830	166	664
4	Corn	237	47	190
5	Grass-pasture	483	97	386
6	Grass-trees	730	146	584
7	Grass-pasture-mowed	28	14	14
8	Hay-windrowed	478	96	382
9	Oats	20	10	10
10	Soybean-notill	972	194	778
11	Soybean-mintill	2455	491	1964
12	Soybean-clean	593	119	474
13	Wheat	205	41	164
14	Woods	1265	253	1012
15	Buildings-Grass-Trees-Drives	386	77	309
16	Stone-Steel-Towers	93	46	47

for testing. In order to compare the performance of different methods, we used three accuracy metrics: OA, AA, and kappa coefficient. All experiments were repeated 10 times and the mean value for each accuracy metric as well as its standard deviation are reported. Experiments are performed using Matlab R2017a on a desktop with an Intel Core i7 3.7 GHz cpu and an NVIDIA GeForce GTX 1080 Ti gpu.

5.3.1 Data Description

Indian Pines hyperspectral dataset is collected over the Indian Pines site in north-western Indiana, USA. AVIRIS with the wavelength range of 0.4-2.5 μm, 224 spectral bands, and image size of 145×145 has been used to gather this dataset. Having removed the 24 water absorption bands, it leaves us with 200 bands in total. Ground truth image of this dataset includes 16 different land cover classes. Figure 5.3 shows the image of band 110

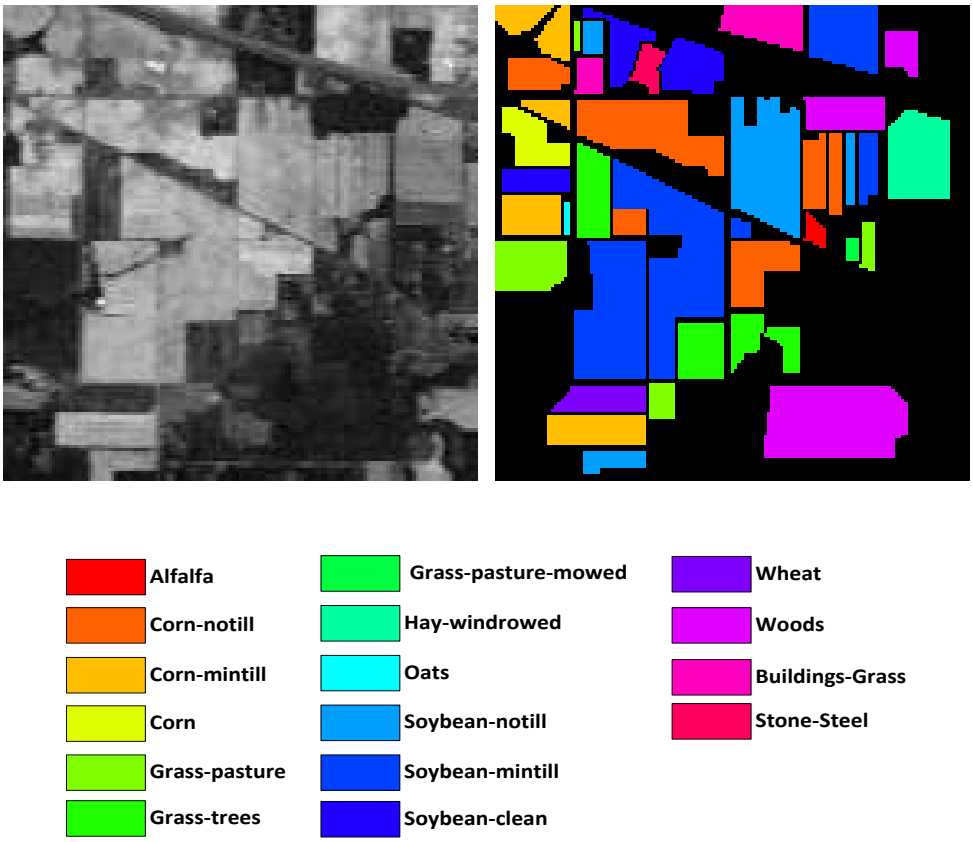


Figure 5.3: Indian Pines dataset. (Left) Image of band 110 and (right) ground truth image.

and the ground truth image of this database.

5.3.2 Performance Evaluation

Usually with increasing the number of training data (original plus virtual samples in our case) classification accuracy increases as well. However, more training data results in longer training time. With creating only 10 virtual samples of each original sample, total number of augmented samples will be 10 times more than the number of original samples. Therefore, high computational resources will be needed to train the classifier using this large number of training data. In our experiments we tried different values for parameter n and to have a trade off between the training time and the accuracy we

chose n=50. For the parameter *shift range*, we need to limit the value of this quantity such that for all classes in the dataset, the virtual samples more or less cover the same reflectance range as the corresponding original samples do. We tested different values for this parameter and experiments showed that value of 100 gives the best amount of shift range for all classes in the dataset. For the deep learning framework we used a three-layer SAE consisting of a sparse autoencoder in each layer with 80 hidden neurons. We used LR classifier which uses softmax function in its output layer activation. This function can be defined as (5.3)

$$S(\boldsymbol{o}_k) = \frac{exp(\boldsymbol{o}_k)}{\sum_{j=1}^{C} exp(\boldsymbol{o}_j)} \tag{5.3}$$

where $\boldsymbol{o}$ and $S(\boldsymbol{o}_k)$ are input vector and the kth element of the output vector of the softmax function. C is the size of the output vector which for a classification problem is equal to the total number of classes. The reason behind using soft-max function is that it guarantees that the sum of all the C entries of the output vector sums to 1, therefore, we can consider the output as a set of conditional probabilities. For example, considering $\boldsymbol{F}$ as an output vector of the last AE in our model and $\boldsymbol{W}$ and $\boldsymbol{b}$ as the weights and biases of the LR layer, conditional probability that vector $\boldsymbol{F}$ belongs to class c is defined as

$$P(y = c|\boldsymbol{F}, \boldsymbol{W}, \boldsymbol{b}) = S(\boldsymbol{F}\boldsymbol{W} + \boldsymbol{b}) = \frac{exp(\boldsymbol{F}\boldsymbol{W}_c + b_c)}{\sum_{j=1}^{C} exp(\boldsymbol{F}\boldsymbol{W}_j + b_j)} \tag{5.4}$$

where the left hand side of the equation represents the probability that feature vector $\boldsymbol{F}$ belongs to class c.

For training each individual AE, we used mean squared error (MSE) loss function with the sparsity and weight regularization terms. For the fine tuning of our SAE network, we used cross-entropy loss function as defined in (5.5)

$$loss = -\sum_{i=1}^{M}\sum_{j=1}^{C} t_{ij} log(y_{ij}) \tag{5.5}$$

where M and C are the total number of training samples and classes, respectively. y_{ij} is the output of the network for the ith training sample for class j, and t_{ij} represents the jth element of the target vector for ith training sample. To test the effectiveness of our

Table 5.2: Class-specific accuracies, OA (%), AA (%), Kappa coefficient, and the test time (s) of the different methods on Indian Pines dataset using 20% of the labeled samples for training.

Class index	RBF SVM	EMAP	Spectral-DAE	PPF-CNN	EMAP-SAE	Proposed method
1	84.35	92.17	80.00	90.43	93.04	99.57
2	81.50	79.84	80.99	91.30	89.04	96.32
3	75.83	91.39	75.20	83.73	90.29	92.33
4	65.42	92.05	64.53	84.05	87.79	93.79
5	92.62	84.66	87.51	93.70	92.93	95.91
6	96.37	96.46	94.28	99.67	97.77	99.64
7	88.57	90.71	77.86	84.28	90.71	100
8	96.52	99.87	95.99	99.45	98.87	99.90
9	79.00	97.00	64.00	68.00	77.00	92.00
10	81.23	74.88	79.41	87.18	89.46	95.75
11	86.25	96.69	83.26	93.66	95.03	97.38
12	80.78	91.81	75.49	90.67	88.06	94.64
13	97.80	97.87	96.22	98.29	96.58	99.63
14	96.13	98.85	92.83	97.74	98.19	99.39
15	60.45	97.96	65.21	71.58	92.56	83.01
16	94.04	100	92.98	99.57	97.44	97.02
OA	85.47±0.34	91.32±0.60	83.39±1.20	91.95 ±0.98	93.28 ±0.45	96.38±0.64
AA	84.80±1.08	92.64±0.93	81.61±2.99	89.58 ± 0.93	92.17 ±1.28	96.02±0.96
Kappa	0.83±0.004	0.90±0.007	0.81±0.014	0.91 ± 0.011	0.92± 0.005	0.96±0.007
Test time(s)	3.20	1.14	0.070	2.41	0.088	0.068

proposed method, we compared it with the following HSI classification methods: RBF SVM, EMAP [37], DAE with spectral features [41], pixel-pair features PPF-CNN [47], EMAP-SAE [48], and 3D-CNN [54]. For the SVM classifier, having performed the 10-fold cross validation to find the best values for the regularization parameter c, and the width

of the Gaussian kernel g, they were obtained as 100 and 1, respectively. With EMAP, we used RBF SVM classifier with c and g equal to 100 and 0.1, respectively. Also, for spectral-DAE, PPF-CNN, and EMAP-SAE methods, we used LR classifier with softmax activation as well.

Table. 5.2 lists the classification accuracies and the test time of these methods. The number of train and the test samples for these methods are the same as our proposed method. As can be seen from this table, our approach outperforms the others in terms of OA, AA, and Kappa coefficient. More specifically, the values obtained for these three metrics using the proposed method are by 3.10%, 3.38%, and 4% higher than the corresponding second highest values in the table. From the class-specific accuracies it can be seen that the proposed method is capable of delivering high classification accuracies for all of the classes and for the majority of them it outperforms the other methods. Also, by comparing our method with 3D-CNN [54] which also uses data augmentation, we observed although this method achieves higher classification accuracies than ours, due to the convolution along spectral bands the computation time is quite high. Table 5.3 compares accuracies and the train and test times of our proposed and the 3D-CNN methods. As can be seen from the table, although 3D-CNN outperforms our method

Table 5.3: Classification accuracy and running time of the proposed and 3D-CNN methods.

Method	3D-CNN	Proposed method
OA(%)	98.53±0.29	96.38±0.64
AA(%)	99.50±0.08	96.02±0.26
Kappa	0.98±0.003	0.96±0.004
Train time (min)	27.04	16.52
Test time (min)	0.88	0.001

in terms of accuracy metrics, it demands relatively longer training and test times. It should be noted that their reported train time is before incorporating virtual samples for training their model.

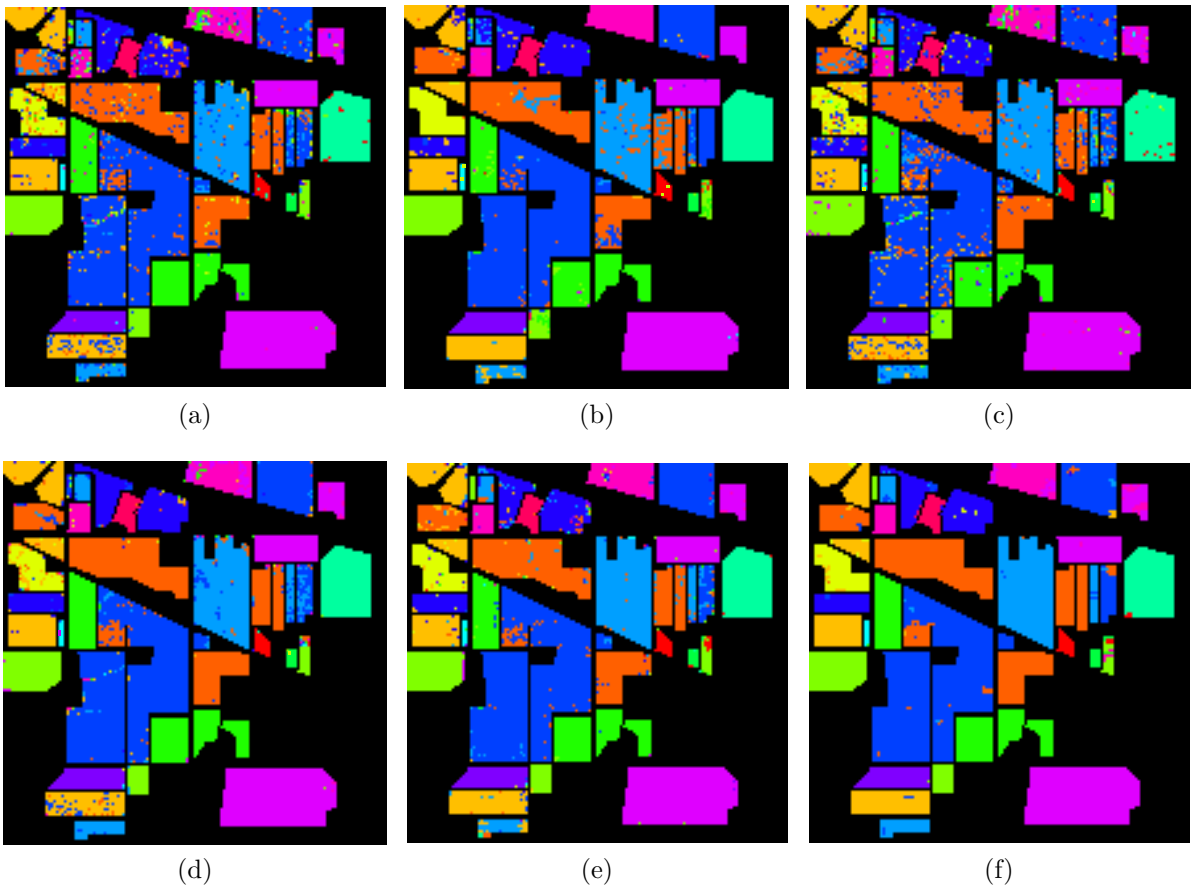


Figure 5.4: Classification maps resulting from different methods. (a) Gaussian RBF-SVM, (b) EMAP, (c) spectral-DAE, (d) PPF-CNN, (e) spectral-EMAP-SAE, and (f) proposed method.

Fig. 5.4 shows the classification maps resulting from the methods experimented in this study. As this figure depicts, our method results in a smoother classification map (i.e., our classification map is less noisy). For example, it can be seen especially for classes *Corn-notill* and *Soybean-notill*, there are far less misclassified pixels in our method's classification map compared to other methods'.

5.4 Conclusion

In this chapter to extract deep spectral features of the Indian Pines hyperspectral dataset, we used a SAE with three layers of sparse AE as the deep learning model. Because of the fact that deep learning models require many training data to be trained effectively and since there are few training samples in the Indian Pines dataset, we proposed a data augmentation method to increase the number of training samples by carefully conserving both the trends of the original samples in each class and the reflectance region covered by them. In our method, new samples are generated out of original data by shifting the spectrum up and down within a reasonable reflectance range. Finally, in order to smooth the output classification map, we used the spatial information of the neighbors of the test pixels in the test step.

Because of its simplicity, our data augmentation method demands a low computational cost. Classification accuracies presented in the experimental results section show the effectiveness of our proposed algorithm and its potential usage in augmenting the remote sensing HSI training samples used for training deep learning models. Also, our algorithm is very fast at the test time which makes it suitable for real time remote sensing HSI classification applications.

Chapter 6

Conclusion

Remote sensing impacts and improves our daily lives in many ways. The diverse data acquisition methods used in this field provide valuable source of information about the objects and phenomena on earth that could hardly been supplied without them. In the past few decades, there have been numerous number of researches working on interpreting the information obtained from remote sensing as accurately as possible to maximize the scientific returns which makes this science to continue and advance. This research work studies the problem of pixel-wise hyperspectral image classification in the remote sensing field.

As the outcome of the most recent type of imaging, hyperspectral images measure the energy of light reflected back from the scene at many contiguous spectral bands providing favorable features for image classification tasks. In an HSI, the type of the objects at different pixels can be determined by analysis of the spectra while spatial information depicts their distribution in the image. Therefore, combining spectral and contextual information can improve the performance of hyperspectral image classifiers. In this thesis, we proposed two frameworks of modeling spectral information together with spatial contextual information to generate spatial-spectral features and employed stacked autoencoder with sparse AEs in its layers to extract deep features of the input

data. A common problem in HSI classification field is the inadequacy of ground truth samples required for training. This problem can be solved to some extent by performing data augmentation. In Chapter 5, we proposed a simple data argumentation technique to boost number of training samples.

Most of the deep learning-based HSI classification approaches use unsupervised dimensionality reduction techniques such as PCA in their pre-processing step. In order to see the effect of using class labels on classification accuracies in the presence of a deep network in the model, in Chapter 3, we proposed a method which combines principal component discriminant analysis (PCDA), a supervised DR technique, with SAE. Such an algorithm exploits class-discriminatory information and the power of a deep neural network in extracting deep features, simultaneously. Results of applying our method, PCDA-SAE on the Indian Pines and University of Pavia hyperspectral datasets demonstrate that it outperforms both PCDA and SAE methods and justifies our intent to such a combination. Also, comparisons with some conventional and recent HSI classification algorithms shows the superiority of our algorithm.

In Chapter 4, we proposed a new distance transform-based spatial feature vector which considers the distance of pixels with respect to the edges in the image as a new feature with the goal of assigning different weights to the adjacent pixels of a target sample. To accomplish this goal, a method to generate the distance transform image of an input HSI is proposed. Furthermore, in order to incorporate more spatial information, we enhanced our spatial feature vector with EMAP features as well. Finally to use class labels in the pre-processing phase, PCDA method was employed prior to form our distance transform-based spatial feature vector. We performed extensive experiments searching in our model's hyperparameter space to find the best values. Having applied our method on three hyerspectral datasets, we came up with the following conclusions: First, assigning different weights to the adjacent pixels according to their proximity to the edges in the image improved classification accuracies. Such an improvement sounds logical because of the fact that not all neighboring pixels belong to the target pixel's category with the

same probability. Next, incorporation of the EMAP features in the spatial feature vector increased the accuracy of classification as well which verified the power of such features in modeling spatial information in hyperspectral images. Finally, as was expected a supervised dimensionality reduction method together with our proposed spatial feature vector enhanced classification accuracies as well. Experimental results also showed that the proposed spectral-spatial feature vectors outperformed some conventional and recent deep learning based HSI classification approaches in terms of accuracy of the segmentation.

Due to the need of deep learning models for enough training data and the challenges in obtaining ground truth images of remote sensing HSI datasets, in Chapter 5, we proposed a data augmentation method to increase the number of labeled samples of each class in the database. Our method generates virtual samples by keeping the pattern of the ground truth samples and the reflectance range covered by them. In this method, we incorporated the spatial contextual information in the test phase. The simplicity of the proposed data augmentation method lets it demand a low computational cost. Classification accuracies obtained by our model show its effectiveness in boosting the training data used for HSI classification purposes.

To summarize, one problem in the field of remote sensing, hyperspectral image classification has been well addressed. however, there are still a lot of work to be done in the future. For example, techniques for spectral-spatial feature generation and data augmentation should be improved and examined on more hyperspectral datasets. Moreover, a simultaneous search in the hyperparameter space should be done with the help of super powerful GPUs. In other words, we should be able to see the effect of each possible combination of the tested values for the model's hyperparameters simultaneously on the classification accuracy.

We can name two potential future applications of the methods proposed in this thesis. First, the proposed approaches can be used in tree type classification of HSI images obtained from remote forests. Furthermore, with some adjustments, they may be applicable for object classification on the surface of other planets rather than earth provided

that ground truth images are available.

Bibliography

[1] K. Tempfli, G. Huurneman, W. Bakker, L. Janssen, W. Feringa, A. Gieske, K. Grabmaier, C. Hecker, J. Horn, N. Kerle, F. van der Meer, G. Parodi, C. Pohl, C. Reeves, F. van Ruitenbeek, E. Schetselaar, M. Weir, E. Westinga, and T. Woldai, *Principles of remote sensing : an introductory textbook*. ITC Educational Textbook Series, Netherlands: International Institute for Geo-Information Science and Earth Observation, 2009.

[2] Z. A. Latif, H. M. Zaqwan, M. Saufi, N. A. Adnan, and H. Omar, "Deforestation and carbon loss estimation at tropical forest using multispectral remote sensing: Case study of besul tambahan permanent forest reserve," in *2015 International Conference on Space Science and Communication (IconSpace)*, pp. 348–351, Aug 2015.

[3] E. Roitberg, V. Barraza, F. Grings, M. Salvia, P. Perna, and M. Barber, "Near real time multisensor algorithm for deforestation alert over the dry chaco forest," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 818–821, July 2018.

[4] G. Katarki, H. Ranmale, I. Bidari, and S. Chickerur, "Estimating change detection of forest area using satellite imagery," in *2019 International Conference on Data Science and Communication (IconDSC)*, pp. 1–8, March 2019.

[5] L. Yue, H. Shen, W. Yu, and L. Zhang, "Monitoring of historical glacier recession in yulong mountain by the integration of multisource remote sensing data," *IEEE Jour-*

nal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 11, pp. 388–400, Feb 2018.

[6] Y. Sun, L. Jiang, L. Liu, Q. Sun, H. Wang, and H. Hsu, "Mapping glacier elevations and their changes in the western qilian mountains, northern tibetan plateau, by bistatic insar," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, pp. 68–78, Jan 2018.

[7] C. Zhou, G. Yang, D. Liang, X. Yang, and B. Xu, "An integrated skeleton extraction and pruning method for spatial recognition of maize seedlings in mgv and uav remote images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 4618–4632, Aug 2018.

[8] A. Singh, "Remote sensing and gis applications for municipal waste management," *Journal of Environmental Management*, vol. 243, pp. 22 – 29, 2019.

[9] Available at https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used.

[10] "HyspIRI mission." `https://hyspiri.jpl.nasa.gov/`.

[11] B. A. Stauffer, H. A. Bowers, E. Buckley, T. W. Davis, T. H. Johengen, R. Kudela, M. A. McManus, H. Purcell, G. J. Smith, A. Vander Woude, and M. N. Tamburri, "Considerations in harmful algal bloom research and monitoring: Perspectives from a consensus-building workshop and technology testing," *Frontiers in Marine Science*, vol. 6, p. 399, 2019.

[12] "HISUI mission." `https://ssl.jspacesystems.or.jp/en_project_hisui/`.

[13] "Advanced spaceborne thermal emission and reflection radiometer." `https://asterweb.jpl.nasa.gov`.

[14] "Landsat." `https://landsat.usgs.gov`.

[15] "Airborne visible/infrared imaging spectrometer." `https://aviris.jpl.nasa.gov`.

[16] I. Dumke, S. M. Nornes, A. Purser, Y. Marcon, M. Ludvigsen, S. L. Ellefmo, G. Johnsen, and F. SÃÿreide, "First hyperspectral imaging survey of the deep seafloor: High-resolution mapping of manganese nodules," *Remote Sensing of Environment*, vol. 209, pp. 19 – 30, 2018.

[17] M. Jiang, F. Cao, and Y. Lu, "Extreme learning machine with enhanced composite feature for spectral-spatial hyperspectral image classification," *IEEE Access*, vol. 6, pp. 22645–22654, 2018.

[18] Y. Guo, H. Cao, S. Han, Y. Sun, and Y. Bai, "SpectralâĂŞspatial hyperspectral-image classification with k-nearest neighbor and guided filter," *IEEE Access*, vol. 6, pp. 18582–18591, 2018.

[19] F. Li, P. Zhang, and L. Huchuan, "Unsupervised band selection of hyperspectral images via multi-dictionary sparse representation," *IEEE Access*, vol. 6, pp. 71632–71643, 2018.

[20] K. Zhou, T. Cheng, X. Deng, X. Yao, Y. Tian, Y. Zhu, and W. Cao, "Assessment of spectral variation between rice canopy components using spectral feature analysis of near-ground hyperspectral imaging data," in *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–4, Aug 2016.

[21] A. Xie, D.-W. Sun, Z. Xu, and Z. Zhu, "Rapid detection of frozen pork quality without thawing by visâĂŞnir hyperspectral imaging technique," *Talanta*, vol. 139, pp. 208 – 215, 2015.

[22] S. Munera, C. Besada, N. Aleixos, P. Talens, A. Salvador, D.-W. Sun, S. Cubero, and J. Blasco, "Non-destructive assessment of the internal quality of intact persimmon using colour and vis/nir hyperspectral imaging," {*LWT*} - *Food Science and Technology*, vol. 77, pp. 241 – 248, 2017.

[23] C. zhang, C. Guo, F. Liu, W. Kong, Y. He, and B. Lou, "Hyperspectral imaging analysis for ripeness evaluation of strawberry with support vector machine," *Journal of Food Engineering*, vol. 179, pp. 11 – 18, 2016.

[24] W. Di, L. Zhang, D. Zhang, and Q. Pan, "Studies on hyperspectral face recognition in visible spectrum with feature band selection," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, pp. 1354–1361, Nov 2010.

[25] D. Manolakis, D. Marden, and G. Shaw, "Hyperspectral image processing for automatic target detection applications," *Lincoln Lab J*, vol. 14, 01 2003.

[26] A. F. Goetz, "Three decades of hyperspectral remote sensing of the earth: A personal view," *Remote Sensing of Environment*, vol. 113, pp. S5 – S16, 2009. Imaging Spectroscopy Special Issue.

[27] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, pp. 1778–1790, Aug 2004.

[28] F. Melgani and L. Bruzzone, "Support vector machines for classification of hyperspectral remote-sensing images," in *IEEE International Geoscience and Remote Sensing Symposium*, vol. 1, pp. 506–508 vol.1, June 2002.

[29] G. H. Halldorsson, J. A. Benediktsson, and J. R. Sveinsson, "Source based feature extraction for support vector machines in hyperspectral classification," in *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, vol. 1, p. 539, Sep. 2004.

[30] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based$k$-nearest-neighbor for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, pp. 4099–4109, Nov 2010.

[31] R. M. Haralick and K. Shanmugam, "Combined spectral and spatial processing of erts imagery data," *Remote Sensing of Environment*, vol. 3, no. 1, pp. 3 – 13, 1974.

[32] P. H. Swain, S. B. Vardeman, and J. C. Tilton, "Contextual classification of multispectral image data," *Pattern Recognition*, vol. 13, no. 6, pp. 429 – 441, 1981.

[33] M. Pesaresi and J. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, pp. 309 – 320, 03 2001.

[34] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, "Classification of hyperspectral data from urban areas based on extended morphological profiles," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 480–491, March 2005.

[35] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using svms and morphological profiles," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, pp. 3804–3814, Nov 2008.

[36] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone, "Morphological attribute profiles for the analysis of very high resolution images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, pp. 3747–3762, Oct 2010.

[37] M. D. Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone, "Extended profiles with morphological attribute filters for the analysis of hyperspectral data," *International Journal of Remote Sensing*, vol. 31, no. 22, pp. 5975–5991, 2010.

[38] G. Camps-Valls, L. Gomez-Chova, J. MuÃśoz-MarÃŋ, J. Vila-FrancÃľs, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 93–97, 2006.

[39] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson, "Generalized composite kernel framework for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, pp. 4816–4829, Sep. 2013.

[40] G. Camps-Valls, L. Gomez-Chova, J. MuÃśoz-MarÃŋ, J. Vila-FrancÃľs, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 93–97, 2006.

[41] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, pp. 2094–2107, June 2014.

[42] Y. Chen, X. Zhao, and X. Jia, "Spectral-spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 2381–2392, June 2015.

[43] X. Ma, H. Wang, and J. Geng, "Spectral-spatial classification of hyperspectral image based on deep auto-encoder," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, pp. 4073–4085, Sept 2016.

[44] X. Ma, J. Geng, and H. Wang, "Hyperspectral image classification via contextual deep learning," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, p. 20, 2015.

[45] W. Zhao and S. Du, "Spectral-spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 4544–4554, Aug 2016.

[46] X. Zhou, S. Li, F. Tang, K. Qin, S. Hu, and S. Liu, "Deep learning with grouped features for spatial spectral classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, pp. 97–101, Jan 2017.

[47] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, pp. 844–853, Feb 2017.

[48] H. Teffahi, H. Yao, S. Chaib, and N. Belabid, "A novel spectral-spatial classification technique for multispectral images using extended multi-attribute profiles and sparse autoencoder," *Remote Sensing Letters*, vol. 10, no. 1, pp. 30–38, 2019.

[49] Z. Lin, Y. Chen, X. Zhao, and G. Wang, "Spectral-spatial classification of hyperspectral image using autoencoders," in *2013 9th International Conference on Information, Communications Signal Processing*, pp. 1–5, Dec 2013.

[50] L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?," in *2011 International Conference on Computer Vision*, pp. 471–478, Nov 2011.

[51] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new bayesian approach with active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, pp. 3947–3960, Oct 2011.

[52] T. Li, J. Zhang, X. Zhao, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *2014 IEEE International Conference on Image Processing (ICIP),*, pp. 1–5, Oct 2014.

[53] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, no. 3, p. 12pages, 2015.

[54] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 6232–6251, Oct 2016.

[55] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 4959–4962, July 2015.

[56] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 1349–1362, March 2016.

[57] B. Pan, Z. Shi, and X. Xu, "R-vcanet: A new deep-learning-based hyperspectral image classification method," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, pp. 1975–1986, May 2017.

[58] L. Shu, K. McIsaac, and G. R. Osinski, "Hyperspectral image classification with stacking spectral patches and convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–10, 2018.

[59] H. T. Chen, H. W. Chang, and T. L. Liu, "Local discriminant embedding and its variants," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 846–853 vol. 2, June 2005.

[60] M. Mohammed, M. B. Khan, and E. B. M. Bashier, *Machine Learning Algorithms and Applications*. Taylor and Francis Group, 2017.

[61] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[62] I. Jolliffe, *Principal Component Analysis*, pp. 1094–1096. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[63] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals Eugenics*, vol. 7, pp. 179–188, 1936.

[64] C. R. Rao, "The utilization of multiple measurements in problems of biological classification," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 10, pp. 159–193, jul 1948.

[65] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. `http://neuralnetworksanddeeplearning.com/`.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[67] D. E. Rumelhart, G. E. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct 1986.

[68] B. Wu, K. Li, F. Ge, Z. Huang, M. Yang, S. M. Siniscalchi, and C. H. Lee, "An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic

modeling for robust speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 1289–1300, Dec 2017.

[69] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised autoencoders," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 2108–2118, Oct 2015.

[70] S. Nagpal, M. Singh, R. Singh, and M. Vatsa, "Regularized deep learning for face recognition with weight variations," *IEEE Access*, vol. 3, pp. 3010–3018, 2015.

[71] "Sparse autoencoder." `https://www.mathworks.com/help/deeplearning/ref/trainautoencoder.html;jsessionid=4d6dead5aac0b2f61758450f8fc6`.

[72] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision Research*, vol. 37, no. 23, pp. 3311 – 3325, 1997.

[73] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, Jul 2006.

[74] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[75] G. Hinton, "A practical guide to training restricted boltzmann machines (version 1)," 08 2010.

[76] P. Soille, *Morphological Image Analysis- Principles and Applications*. Springer, 2003.

[77] E. J. Breen and R. Jones, "Attribute openings, thinnings, and granulometries," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 377 – 389, 1996.

[78] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Transactions on Image Processing*, vol. 7, pp. 555–570, April 1998.

[79] C. Lee and D. A. Landgrebe, "Analyzing high-dimensional multispectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 31, pp. 792–800, Jul 1993.

[80] C.-I. Chang, Q. Du, T.-L. Sun, and M. L. G. Althouse, "A joint band prioritization and band-decorrelation approach to band selection for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, pp. 2631–2641, Nov 1999.

[81] L. O. Jimenez and D. A. Landgrebe, "Hyperspectral data analysis and supervised feature reduction via projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, pp. 2653–2667, Nov 1999.

[82] L. M. Bruce, C. H. Koger, and J. Li, "Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, pp. 2331–2338, Oct 2002.

[83] M. Imani and H. Ghassemian, "Principal component discriminant analysis for feature extraction and classification of hyperspectral images," in *2014 Iranian Conference on Intelligent Systems (ICIS)*, pp. 1–5, Feb 2014.

[84] Available at https://www.csie.ntu.edu.tw/%7Ecjlin/libsvm/.

[85] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "Svm- and mrf-based method for accurate classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, pp. 736–740, Oct 2010.

[86] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Upper Saddle River, N.J.: Prentice Hall, 2008.

[87] C. R. Maurer, Rensheng Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 265–270, Feb 2003.

Curriculum Vitae

Name:	Hadis Madani
Post-Secondary Education and Degrees:	The University of Western Ontario London, ON, Canada 2015 - 2020, PhD in Electrical Engineering University of Guilan Rasht, Guilan, Iran 2010 - 2013, M.Sc. in Electrical Engineering University of Guilan Rasht, Guilan, Iran 2004 - 2009, B.Sc. in Electrical Engineering
Honors and Awards:	Western Graduate Research Scholarship 2015-2019
Related Work Experience:	Research and Teaching Assistant The University of Western Ontario 2015 - 2019

Publications:

- Madani, H., and K. McIsaac. Spectral perturbation method for deep learning-based classification of remote sensing hyperspectral images. In Image and Signal Processing for Remote Sensing XXV, vol. 11155, pp.260−269, International Society for Optics and Photonics, SPIE, 2019.

- Madani, H., and McIsaac, K. Distance transform based spectral-spatial feature vec-

tor for hyperspectral image classification with stacked autoencoder. IEEE Access. Under review.

- Madani, H., and McIsaac, K. Hyperspectral Image Classification Using PCDA and SAE. Computers & Geosciences. Under review.